

Creating a mobile VR interactive tour guide

Robert Lee Seligmann

Bachelor's Thesis
Degree Program in BIT
2018



Author(s) Robert Lee Seligmann	
Degree program Bachelor of Science, Business Information Technology	
Report/thesis title Creating a mobile VR interactive tour guide	Number of pages and appendix pages 74 + 3
<p>Virtual Reality is a concept that has existed since 1960. However, it wasn't very popular in the consumer segment until these recent years. Since 2015, many Virtual Reality headsets arrived on the public market and the development of immersive experiences has opened the gateway to many new opportunities. As part of this work, we will develop a VR application and focus our research on the history of this technology and its presence in the industry.</p> <p>A good VR experience usually requires powerful hardware and equipment, but with the improvement of smartphone technology, we are now able to create VR application that is playable on most mobile phone. Their hardware and screen resolution are good enough to enjoy a basic VR experience.</p> <p>This thesis will be about developing an application that will allow the users to visit certain points of interest within a city. Inside the virtual world made using 360° footages, the users will be able to select the point they wish to visit with their gaze and will be able to obtain information about the place.</p> <p>The application will be created using the Unity game engine which is used to develop 2D/3D games as well as VR and AR applications. It supports C# as a scripting language and is the easiest and most common tool to develop a VR application. As we don't have the necessary hardware to run a high-end VR gear, we will be using a smartphone and a Google Cardboard.</p> <p>The testing phase is conducted with 11 people who will test the application and answer a survey about the experience and the usability of the product.</p> <p>The first part of this thesis will include the basic knowledge of virtual reality as well as its uses in the industry. Then, the second part will discuss the development of the application using Unity 3D.</p> <p>It is concluded with a discussion regarding the research that was carried out and the overall product development process.</p>	
Keywords Application, Virtual Reality, Tourism, Mobile, Technologies	

Table of contents

1	Introduction	1
2	Research question	2
2.1	Objective of the project	2
2.2	Scope of the project	2
3	Background study	3
3.1	Virtual Reality.....	3
3.1.1	How virtual reality works.....	3
3.1.2	Virtual 3D environment.....	5
3.1.3	360° video.....	6
3.2	Augmented reality	7
3.3	Headset	8
3.3.1	What is a virtual reality headset.....	9
3.3.2	Cheap headset.....	9
3.3.3	Mid-range headset	10
3.3.4	High-end headset.....	11
3.4	Headsets sales	17
3.5	Types of interactions.....	18
3.5.1	Navigation.....	18
3.5.2	Selection	20
3.5.3	Manipulation.....	20
3.6	Virtual Reality usage	20
3.6.1	Entertainment.....	20
3.6.2	Military	21
3.6.3	Healthcare.....	21
3.6.4	Education.....	22
3.6.5	Engineering.....	22
3.6.6	Tourism.....	23
4	Development framework	24
4.1	Unity 3D.....	24
4.2	Unreal Engine	24
4.3	CryEngine	25
4.4	Selected framework	26
5	Empirical part.....	27
5.1	Interactions	27
5.1.1	Types of interactions.....	27
5.2	Scenario.....	28
5.3	Prototype development	30

5.3.1	Plugins used	30
5.3.2	Equipment.....	32
5.4	Unity setup.....	32
5.4.1	Creating and setting up a scene.....	32
5.4.2	Video player	34
5.4.3	Building the application to smartphone.....	36
5.5	Unity development	37
5.5.1	Blink effect	37
5.5.2	Level manager	39
5.5.3	Interactive button.....	40
5.5.4	Menu.....	45
5.5.5	Multi-language	55
5.5.6	Voice command	57
5.6	Results.....	58
5.6.1	Approach	58
5.6.2	Test users	59
5.6.3	Questions.....	59
5.6.4	End results	60
6	Discussion.....	62
6.1	Summary	62
6.2	Conclusion	62
6.3	Quality of the results	63
6.4	Problems encountered.....	63
6.5	Evaluation of the thesis process.....	64
6.5.1	Methodology	64
6.5.2	Evaluation of self-learning.....	64
6.6	Project management.....	64
6.7	Ethical viewpoints	65
6.8	Further development.....	65
7	Table of Figures	66
8	References.....	68
	Appendices.....	75
	Appendix 1. Results of the survey	75
	Appendix 2. Statistics of the results.....	76
	Appendix 3. Unity project	77

Abbreviation

HES-SO	Haute école spécialisée de Suisse occidentale
VR	Virtual Reality
AR	Augmented Reality
MR	Mixed Reality
XR	Cross Reality
DOF	Degrees of freedom
SDK	Software Development Kit;
C#	C Sharp
MP3	MPEG-1/2 Audio Layer III
3D	Three-dimensional
Fps	Frames per second
HMD	Head-mounted display
CES	Consumer Electronics Show
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
PS4	PlayStation 4
UHD	Ultra-High Definition
OLED	Organic Light-Emitting Diode
HP	Hewlett-Packard
MVR	Windows Mixed Reality
PTSD	Post Traumatic Stress Disorder
FIVE	Ford Immersive Vehicle Environment Lab
RGBA	Red Green Blue Alpha
UI	User Interface

1 Introduction

Since 2015, multiple virtual reality headsets were announced to enter the market at different specifications and for different prices. Many headsets were released in the end of 2017 and more are scheduled to release in 2018. To be able to use most of the advanced headsets, they require a powerful computer to run them at a satisfying setting; more precisely, the requirements are: a screen with a good resolution, a fast CPU and a powerful GPU. Not everybody has a computer, or, if they do have one, it is often not powerful enough to run it but, fortunately, many smartphones possess all three and are sufficiently powerful to run a simple VR application. The only thing needed specifically for smartphones is that they possess a gyroscope and an accelerometer, which is the case for most recent phones.

This has made the development and usage of VR more interesting as well as more accessible for everyone. One of the common misconceptions about VR is that it is solely used for gaming; there are many areas where virtual reality and augmented reality can be effective such as (Mark, 2017):

- Education.
- Healthcare.
- Military.
- Tourism.
- Engineering.
- Other.

The result of this thesis is a fully developed virtual reality application prototype which will serve as a base for future VR mobile applications. The application will let the user dive into a virtual world and visit the places that have been predefined and learn more about the said place. This application could be a tool for tourism advertisement if created properly.

This report will describe the technologies used in order to create the final product. All the tools and equipments will be listed in a later chapter and source code will be included and explained. The end product is an Android application that will work with any compatible Android smartphone (minimum API level 4.4 "Kit Kat") and mobile headset.

First, we will define the objectives and the scope of this project, then, we will look into the background of the technology to have a better understanding of how it works and how it is currently used. Then, we will discuss about the product and the tools used during its development. Finally, the application will be tested and the feedback will be analyzed to evaluate the state of the product.

2 Research question

2.1 Objective of the project

The objective of this project is to create a mobile VR application that could be reimplemented for professional usage. Some features should be thought beforehand and added to avoid as much as possible the risk of motion sickness. The design of the application also needs to be friendly and give enough feedback so that the immersion is not broken by asking someone how to use it.

This is a first step to develop, in the future, a full 3D virtual reality application using an HTC Vive. This will serve as an introduction to learn the bases of Unity 3D as most VR applications are created using this framework.

2.2 Scope of the project

There are different types of possible interactions in the virtual world, we will discuss many of them in a later chapter (3.5). Using only a smartphone and a headset, we are limited to using the gaze input; this means that the user will be able to interact with the world by staring at an interactable object. This application will also integrate some basic voice commands to launch a specific action.

Inside the application, as a user, we will be able to select which path we wish to visit (the end application will have a path for Finland and one for Switzerland). After selecting a path, they will navigate from scene to scene and, inside each scene, they will be able to receive intel about the place that they are currently seeing. The application will support both English and French but should be modifiable to integrate more languages in the future.

This application will provide two ways to present information:

- Audio source (MP3).
- A raw text.

This will allow us to check, during the testing phase, which of them makes the user feel more immersed. To be able to see the difference between a 360° VR video and 3D virtual environment, we will create a small virtual world where the user can see computer generated objects. This will be used as the 3D environment for the main menu.

3 Background study

There are different technologies available to create an immersive experience. In this chapter will discuss mostly about virtual reality, but we will also take a brief look at augmented reality.

3.1 Virtual Reality

According to Oxford dictionaries, virtual reality is defined as the following:

“The computer-generated simulation of a three-dimensional image or environment that can be interacted with in a seemingly real or physical way by a person using special electronic equipment, such as a helmet with a screen inside or gloves fitted with sensors.” (Virtual reality, 2018)

Other sources may have a different definition but there are a few important aspects that can be retained from this definition. The first is the aspect of simulation, the computer will generate the 3D environment for the users to immerse themselves in; a virtual world is, however, not only limited to those 3D creations, now, it is possible to capture 360° videos with the adequate camera and display the video around the user as if they were in the scene. The second point is the interactions that the user can perform. The level of interactivity depends on the equipment and the type of simulation, a 3D world has fewer limitations than a 360° video. This definition does not really include the senses of a person, we experience the world through our senses and the more it is considered in an immersive experience, the more the virtual world will feel real.

3.1.1 How virtual reality works

This chapter will discuss the necessary components to run any VR applications.

3.1.1.1 A stereoscopic display

A stereoscopic display allows the user to perceive depth in an image as well as other depth-based effects such as parallax, converging lines and shading. This is done by presenting a different view of the world to each eye. Some headsets have two different screens, one for each eye, and others such as headsets using smartphones use only one and display both views on the same screen.

Another important part of the headset are the lenses between the eyes and the screen. If we take our phone close to our eyes, the content displayed on our phone will be blurry. All objects closer than ~25 cm from the eye will be perceived as blurry (punctum proximum). The lenses are here to avoid that blurriness, they focus and reshape the picture for each eye and create a stereoscopic 3D image. With those lenses, the image will look like it is further away. Once the screen is displaying our pictures, our brain will treat all the data provided by both pictures to create only one (tpeimageanimee, 2016).

Figure 1 image shows the two different views in a stereoscopic image.



Figure 1 : Stereoscopic 3D image (tpeimageanimee, 2016)

3.1.1.2 Movement-based sensors

These different sensors, such as a gyroscope and an accelerometer, are used to track the movement of the user's head. The more expensive headsets include those sensors and some additional ones like a magnetometer, but for cheaper ones relying on a smartphone, it will depend on the phone. The gyroscope tracks our orientation which is essential if we want to look around the virtual world and the accelerometer measures the speed of our movement.

Regarding the movement, there are 6 movements, also called degree of freedom, that the user can carry out. They can be split into two categories: rotational movement (pitch, yaw, and roll) and translational movement (left/right, forward/backward and up/down). Having all 6 DOF makes for a more immersive VR experience (Degrees of freedom, 2016).

3.1.1.3 Input device

This is how the user will be able to interact with the virtual world, it can be a keyboard or even a mouse. Some headsets have interactable buttons on the side to fetch the user's input. The lowest level of input is our gaze, this allows us to interact in the virtual world by looking at a virtual object. There are many ways to interact with the virtual world, we will go through some of them in a later chapter (3.5).

3.1.2 Virtual 3D environment

A virtual environment is a computer-generated world. The first virtual 3D environments dates from the end of the 60s with the work of Ivan Sutherland, he created what is widely considered to be the first head-mounted display for an immersive simulation (Computer graphics, 2018).

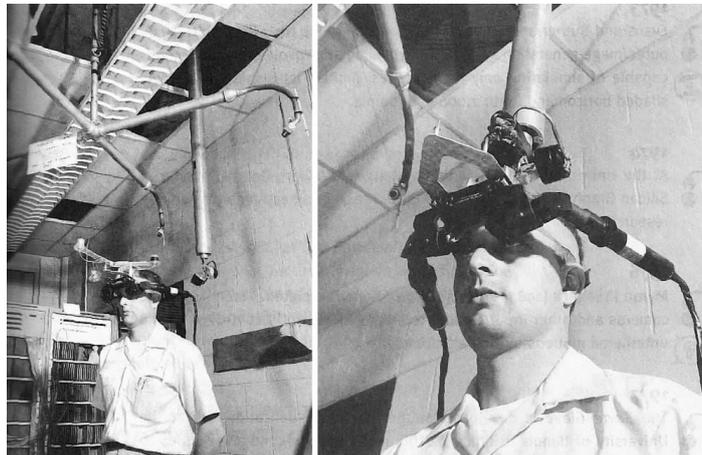


Figure 2 : Sutherland's HMD, The Sword of Damocles (Computer graphics, 2018)

However, the computers of the time weren't powerful, in fact, the only thing rendered was the wireframes of the environment.

Until 2000, there have been many attempts to popularize VR as the technology improved over the years, but none really succeeded. Even the video game giant Nintendo released Virtual Boy in 1995, the first immersive virtual console. They took an innovative risk and unfortunately, it was a failure. The players were complaining that the system was too expensive, had unimpressive 3D graphics and raised concern for the user's health. This failure, among those of other businesses, has greatly slowed down the research on virtual reality (Edwards, 2015).

The interest of the general public started to rise again when, in 2012, American entrepreneur Palmer Luckey created a crowdfunding campaign on Kickstarter to finance the development of his VR headset the Oculus Rift (3.3.4.1), the company was later bought by Facebook in 2014 for 2 billion American dollars. That has initiated many companies to enter the market of virtual reality and offer their own headset.

Nowadays, the problem isn't the quality of the virtual 3D environment. The technology of 3D computer graphics has greatly evolved, it is now possible to create a 3D environment near reality; the limit at the moment is the quality of the screen resolution which can be improved, and the usability (advanced headsets require a lot of setup and restrain the user's movements because of all the cables).

3.1.3 360° video

360° or spherical videos are footages where all points of view are recorded at the same time. It was made popular in 2015 when YouTube supported this format. They provide two viewing methods:

- In the browser on a PC: this method is not immersive, the viewer uses the mouse to move the field of view of the video.
- YouTube application on smartphones: this method can be used with a cardboard or other headset for an immersive experience. A headset is, however, not mandatory, they can view a 360° video as a single view and tilt the phone to change the point of view (Admin_renaud, 2016).

To record a 360° footage, we need a special type of camera also known as 360° cameras. There are many 360° cameras available on the markets, however, not all offer true 360° coverage. Usually, they have multiple lenses to be able to cover every direction. Some solutions like the GoPro omni consist of attaching multiple cameras to cover every direction. The price of the cameras can range from hundreds of euros (e.g. Samsung Gear 360) to thousands of euros (e.g. Nokia OZO).

The main advantage of 360° footages is the ability to create immersive content quickly and simply. Compared to a 3D virtual world where we must create the environment, with a 360° camera, we only need to place it in the adequate position and press record. This allows the user to be placed in the center of the action and get the same perspective than the participants in the video. For example, cosmonauts Fyodor Yurchikhin and Sergey Ryazansky captured some 360° videos during their spacewalk to show what they see on a daily basis during a mission (Reichhardt, 2017).



Figure 3 : Example of a flat 360° picture

However, there are many limitations to 360° footage. First, the quality of the footage which is limited to the lenses on the camera and the compression of the video file. 4K cameras offer tremendous quality when taking standard pictures but that resolution for a spherical video is not impressive (Inc., 2016). The human eye can detect 60 pixels per degree at the fovea (back of the retina where the visual acuity is the sharpest), in comparison, the more advanced headsets have approximately 11 pixels per degree. This causes the screen to look pixelated to the user. As screen technology improves, it will get closer to eye resolution and provide photorealistic experience (Boger, 2017). Secondly, in a 360° video, the users aren't active, they merely watch the video that plays in front of them without being able to interact with it. This makes the experience less immersive, this is also called passive VR.

As it is easier to achieve, 360° footage will be our primary resources to offer VR experiences in our application, making a 3D environment takes a lot of time and is a tedious process.

3.2 Augmented reality

Sometimes confused with virtual reality technology, AR offers a different kind of immersion:

“An enhanced version of reality where live direct or indirect views of physical real-world environments are augmented with superimposed computer-generated images over a user's view of the real-world, thus enhancing one's current perception of reality.” (The Ultimate Guide to Augmented Reality (AR) Technology, s.d.)

Where virtual reality transports the user in the virtual world, augmented reality brings virtual objects and overlays them on top of the real world. The user experiences a new and improved reality. The level of immersion is then lower than VR, but the purpose of augmented reality is not to simulate a new reality but to “augment” the world we live in.

AR can be done with our smartphone; with the correct application, the camera will open and overlay the objects on the screen of our phone. Some augmented reality glasses are available (e.g. Microsoft’s HoloLens) but are more expensive than most virtual reality headsets. During CES 2018, more affordable AR glasses were announced to release during this year (Maubon, 2018).

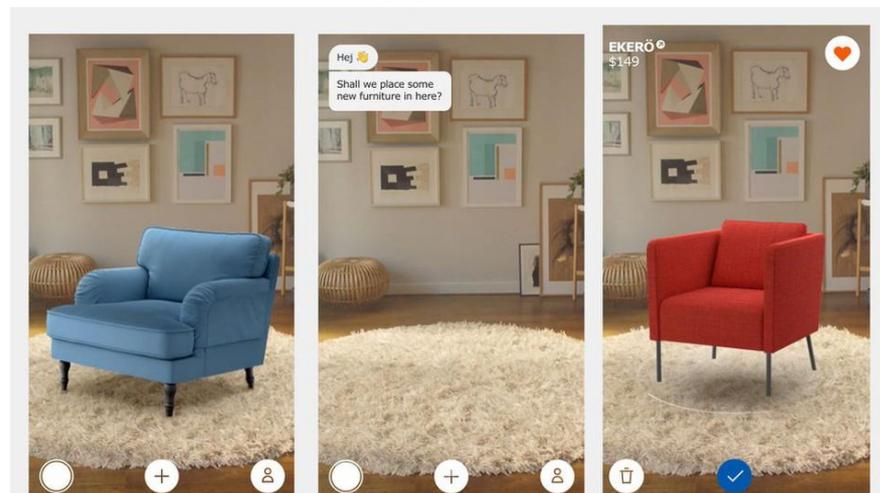


Figure 4 : Example of augmented reality (Bell, 2017)

Many businesses started using AR technology more seriously. Apple release their augmented reality framework (ARKit) for developers and iPhone users which was used by IKEA to develop an AR application that allows their customer to place and preview a piece of furniture in their home before they buy it (Bell, 2017).

The investment in VR / AR has increased significantly over the last few years, breaking records last year as start-ups raised over 3 billion. 1,5 billion was raised during Q4 (Digi-Capital, 2018). A chart of the different industries and investors can be found on Digi-Capital’s website.

3.3 Headset

This chapter will describe some of the different types of virtual reality headset found on the market.

3.3.1 What is a virtual reality headset

VR headsets or Head Mounted Display allows the user to be immersed in a virtual environment or 360° video. The components inside the headset may differ from one another, Figure 5 shows the different components found in the Oculus Rift (How It Works Team, 2016).



Figure 5 : Teardown of the Oculus Rift (How It Works Team, 2016)

Headsets usually require an additional device to run it, cheaper ones require a smartphone and more expensive ones require a computer or a gaming console. Autonomous headsets are slowly entering the market, they are self-powered and do not require any other device to be functional.

3.3.2 Cheap headset

These are the cheapest yet most available headsets on the market. The design is simple and the price is low, mainly because it is made from nothing but a pair of lenses and a sheet of cardboard or another material like plastic or aluminum. As cheap as they are, they come with the hidden cost of a smartphone to be able to run a VR experience. However, some may not consider this a cost as more than $\frac{3}{4}$ of Americans owns a smartphone which is most likely compatible with the Google Cardboard (Robertson, s.d.).

The most notorious one is the Google Cardboard released in 2014 by Google. A Google Cardboard retails at around 15 € on Google's online store.



Figure 6 : Google Cardboard V2 (Google Cardboard V2, s.d.)

These types of headset usually do not have any hardware in it; some, like the Google Cardboard 2.0, have a small button that emulates the action of the user clicking the screen or others have a hole for the users to let their finger in and tap the screen. They are not very comfortable to wear because of the material and often come without any straps to hold it on our head. Those who do not have a button relies on a gaze interaction (looking at an object to click it).

3.3.3 Mid-range headset

They are more expensive than a Google Cardboard but offer additional features such as: tracking sensors, built-in controls, focus wheels and for some, their own screen. The most notorious one is the Samsung Gear VR. There are other options as well, like the Zeiss VR One or the Homido V2. Their designs are, however, less sophisticate than the Gear VR. Samsung's headset retails at 149,90 € (verkkokauppa, price checked on 16.04.2018).



Figure 7 : Samsung Gear VR with controller (Gear VR, s.d.)

Like for the Cardboard, a phone is required, but with the Gear VR, it is limited to certain model of Samsung smartphones. Mid-range headsets are more comfortable and offers better experience while still being portable.

The two previous types of headsets are what is known as **passive** headsets. Usually, they do not require a computer to power it and only need a smartphone. They allow the user to immerse themselves in the virtual world, but they are not in control of the experience, it is like watching a movie with a bigger field of view (Pete, 2016). This is the case with most

headset depending on a phone, however, Samsung and Google have broken that rule by pairing a controller to the headset. Samsung's controller, designed with the collaboration of Oculus, provides additional interactions. This controller is also virtualized in the virtual world, it can take the shape of the controller to simply interact with the menu but can also take the shape of another object such as a gun for shooting a target or a brush to paint in 3D (Hardawar, 2017).

Some mid-range headset now offers the ability to experience **active VR** or **interactive VR** which significantly makes them more interesting.

The primary concern with cheap and mid-range headset is that it may cause motion sickness. With the gyroscope, they are able to track the rotation of our head, but the headset does not have any sensor to track our positional tracking (the spatial movement of our head) so only offers 3 Degrees of freedom. To be able to have all 6 DOF, they need external or internal cameras which is only a feature for high-end headsets.

For our application, we use a custom headset bought during the World VR Forum in Crans-Montana, Switzerland. This headset does not offer any additional input but has two focus wheels to adjust the distance of the lenses and the gap between the eye.

3.3.4 High-end headset

These offer the best VR experience, they are not powered by a phone but by a computer or a gaming console. We will discuss about the main headsets: HTC Vive, Oculus Rift, PlayStation VR and the recent Windows Mixed Reality series. They all offer **active VR** or **interactive VR** as well as 6 DOF.

They are significantly more expensive than other headsets and, to run it, they need a computer or a gaming console. By being powered by such powerful hardware, high-end headsets offer more sophisticated features like motion tracking, higher screen resolution and better graphics. The headset itself is more comfortable and does a better job at blocking outside light. As mentioned before with the mid-range headsets, they have external hardware to follow the user's positional tracking, so the users are less plausible to suffer from motion sickness. The negative point is that compared to the other headsets, they need to be constantly connected to the computer, meaning that the user is tethered to the computer. The user will be able to move freely but the cables will often tangle around their feet and sometime restraint their movement.

Before diving into the headsets, we will discuss about their hidden fees. For the HTC Vive and the Oculus Rift, they both run on computers and the PlayStation VR runs on any model of the PlayStation 4. The hardware required in the computer to run the headset is specific and no average computer has the hardware needed, the main piece of equipment is the graphics card which is used to render the virtual world, other hardware like the CPU and RAM must also meet the requirements. These are quite expensive components and to be able to have a smooth experience, it is always recommended to get hardware that is better than the minimum required. The following image shows the requirement for a desktop computer (Nvidia):

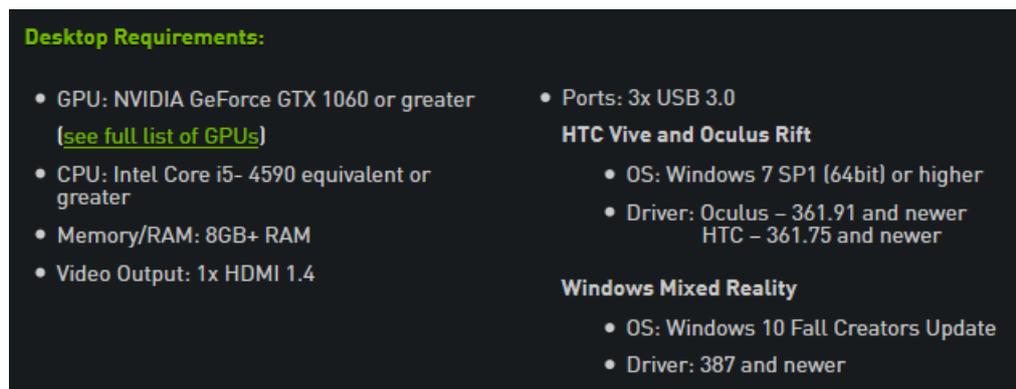


Figure 8 : Desktop requirements for VR (Nvidia)

The prices for the required parts are the following (based on verkkokauppa.com, checked on 28 February).

- | | |
|------------------------------------|----------|
| • GPU: NVIDIA GeForce GTX 1060 | 406,90 € |
| • CPU: Intel Core i5- 4590 | 212,90 € |
| • RAM: Kingston HyperX FURY (8 GB) | 100,90 € |

Total	720,70 €
--------------	-----------------

There are more parts needed for a complete computer, but we can see that a computer to run it is quite expensive, the author's personal home computer setup which is a bit better than the requirements, costs more than 2'000 €.

The advantage of the PlayStation VR is that the PlayStation console is more affordable and is already owned by many. The basic PS4 is discontinued but a second hand version can be bought at GameStop.

Now that we have learned about what is needed to run a high-end headset, we can now look at the different popular headsets on the market.

3.3.4.1 Oculus Rift

This headset is made by Oculus VR and was a result of a crowdfunding campaign on Kickstarter in 2012. The goal of the campaign was to raise 250'000 \$ and ended up raising 2'437'429\$ with 9'522 backers (Oculus, 2016). This project was meant for developers mostly and released a first version of their development kit (DK1) in late 2012; after getting enough feedback, they started working the second version of their kit (DK2) for 2013. They improved the design, the specs, and most of all, added positional tracking. During 2013, more Oculus Rifts were sold, YouTube saw an increase in Oculus Rift reaction videos including YouTube's biggest entertainer PewDiePie who made several videos using the DK1; having the most subscribers on the platform, his videos might have helped to put virtual reality in the front scene for a while. In 2014, Oculus was bought by Mark Zuckerberg's Facebook, this was seen as a betrayal by some who funded the crowdfunding campaign (Oculus Rift History – How it All Started, 2015).



Figure 9 : Oculus Rift with touch controllers (Oculus Rift, s.d.)

The Oculus Rift now cost 509,90€ (verkkokauppa, price checked on 16.04.2018), the original price was higher but was reduced as it was an entry barrier to many people desiring to adopt VR technology. That reduction was also due to new headsets entering the market. The 90 Hz display has a screen resolution of 2160x1200 pixels. The Oculus Rift comes with a headset, 2 Oculus touch controllers, 1 remote controller and 2 Oculus sensors. The Oculus touch controllers allow the user reach out into the virtual world as if it was their hands and interact with virtual objects, increasing the feeling of immersion (Nield, 2016). The Oculus sensors are used for the positional tracking of the headset and controllers. We can download VR games and applications from the Oculus store.

3.3.4.2 HTC Vive

This headset was released in 2016, following a partnership between the smartphone company HTC and the gaming company Valve. The HTC Vive costs 699,90€ (price checked on 16.04.2018) and comes with a headset, 2 Vive controllers, 2 base stations and a link

box (to connect to the computer). The price was also reduced like for the Oculus Rift and is now even cheaper after the release of the HTC Vive Pro. The headset has a 90 Hz display with a screen resolution of 2160x1200 pixels, 37 sensors for seamless movement and a front-facing camera for safety measure; that camera allows the user to, in two-clicks, switch to a view of the real world (Swider & Fitzsimmons, 2018). The partnership with Valve gave HTC a head start as, Valve's gaming platform Steam counts more than 17'000'000 users (Steam & Game Stats, 2018). Valve opened a new section (SteamVR) for VR games and applications, and some are now also playable on Oculus Rift. In February 2018, Oculus has taken the lead in a survey that details the Steam VR headsets usage by holding a user base of 47.31 % while HTC owns 45.38 % (Koolonavich, 2018).



Figure 10 : HTC Vive (Verkkokauppa, s.d.)

The base stations have a similar function than the Oculus sensor except that they work with a different technology. Oculus Rift's sensors use a camera to track our position while HTC Vive's base station uses laser tracking which is more accurate. The base stations (also called Lighthouse) contain infrared laser emitters that rapidly rotate to send a laser beam across the room and LEDs that flash 60 times each second; the invisible light will be caught by the headset's sensors which will then calculate the user's position and orientation. (Buckley, 2015). This system is, however, not appropriate near windows or near broad daylight as it might reflect or interfere with the infrared tracking.

With 2 base stations, HTC is able to provide "room-scale" immersion; a normal VR application is usually experienced sitting down or standing up, but with room-scale, the user has a playing field where they can walk around freely in both the real and virtual world, increasing the immersion of the user.

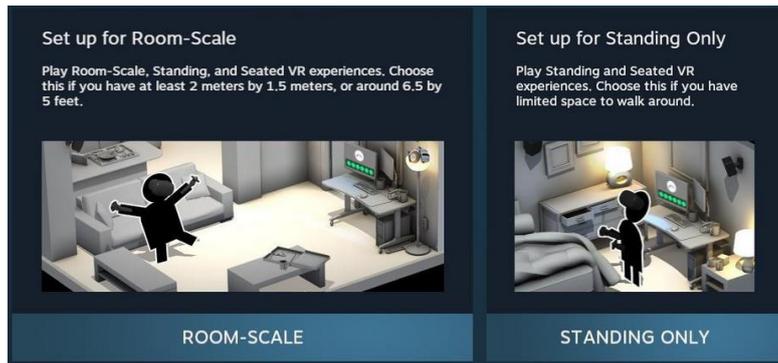


Figure 11 : Room-scale setup for HTC Vive

Oculus offered only one sensor at first which only allowed standing or seated experience then added another sensor to the package to also offer room-scale experience. However, HTC Vive's two tracking stations still provide a bigger play area than the Oculus Rift, even with 3 sensors connected. The following image displays the different play areas for both headset (Lang, 2016).

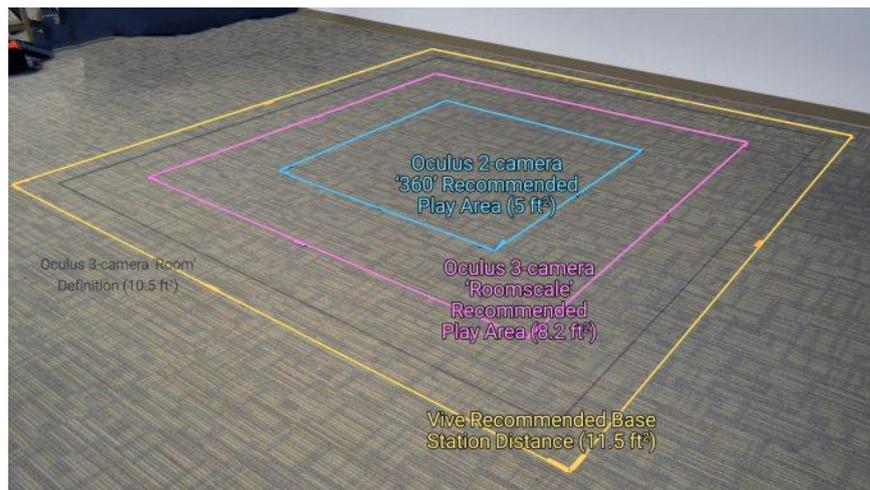


Figure 12 : Play area for HTC Vive and Oculus Rift (Lang, 2016)

3.3.4.3 PlayStation VR (PSVR)

Initially called Project Morpheus, this headset is made by Sony Interactive Entertainment for its latest gaming console the PlayStation 4. Unlike other headsets, this one isn't powered by a computer, but works with any of the fourth generation PlayStation. The PSVR is a very affordable introduction to immersive reality. Released in 2016, the headset costs 315.90€ (price checked on 16.04.2018) and comes with the headset, a small processor unit to be able to connect both a headset and a TV to the PlayStation at the same time, a pair of stereo headphones, a camera and a PlayStation VR Worlds game. The camera serves the same functionality than the Oculus Rift's camera, it is used for positional track-

ing; as it has only one camera, the PSVR do not offer room-scale experiences. The original PS4 controller can be used to play most virtual reality games, but for a better VR experience, it is recommended to purchase the PlayStation Move controllers which exist since the PlayStation 3. Mandatory for some PSVR games, these controller offers better immersion and provide functions similar to the Oculus Touch controller.



Figure 13 : PSVR with a PS4 Pro (Verkkokauppa, s.d.)

Compared to the other headsets, the PSVR is less powerful as it runs on a PlayStation 4. The screen resolution is also lower at 1920x1080 pixels, but the display has a better refresh rate of 120 Hz. This makes the experience more comfortable to the eyes and decreases the risk of motion sickness. The screen also has a better red, green, blue ratio so the blackness of the gaps between the pixels is less visible (Peckham, 2016).

3.3.4.4 Windows Mixed Reality (WMR)

Windows Mixed Reality is a catalogue of headsets resulted from a partnership between Microsoft and other IT manufacturers. All those companies designed their own headset according to Microsoft's specific requirements but are using the same Microsoft motion controllers. Microsoft chose Mixed Reality as a name because they believe virtual reality and augmented reality will eventually blend together (Warren, Microsoft's Windows Mixed Reality: everything you need to know, 2017). The term XR, or Cross Reality, is also used in the industry to refer to all three: VR, AR, MR. At the moment, these headsets focus heavily on VR experiences but will eventually integrate more AR experiences in the future.

They are currently 6 headsets under the Windows Mixed Reality brand (Microsoft, s.d.), they are respectively built by Acer, Dell, HP, Lenovo, Asus and Samsung; released in end 2017 or beginning 2018, their prices are different but ranges around the price of the Oculus Rift or the PSVR.



Figure 14 : All 6 WMR headsets (Microsoft, s.d.)

Compared to the Oculus and the Vive, the WMR headsets use two onboard cameras to determine the user's precise position in the environment (Aaron, Zeller, & Wojciakowski, 2017). This technology is similar to Microsoft's HoloLens AR headsets, which was willingly selected to be able to provide a starting point for both VR and AR development in a single headset. These headsets do not use an external camera and depend solely on the two built-in cameras and built-in sensors (called inside-out tracking), because of that, the two controllers can only be tracked if they are in the field of view of the camera; if the user reaches behind their back, the controllers will not be tracked as opposed to the HTC Vive. These headsets require less equipment which also makes it more portable and faster to setup. The specification for the computer is not as high as the one needed for the Oculus Rift and HTC Vive which is an advantage.

3.4 Headsets sales

The VR market is growing slower than some might have hoped but the sales are slowly improving thanks to the decision of HTC and Oculus VR to decrease the price of their headsets. The VR customer segment is very reliant on the price and their price cut has proven to be a successful decision. For the first time, VR headset sales exceeded 1 million units sold in a single quarter (Q3 2017).

Having the lowest price and being the most accessible, the PlayStation VR took almost a majority of the sales, shipping more than 490'000 units. Seconded by Oculus VR who shipped 210'000 Oculus Rift. HTC took third place by shipping 160'000 HTC Vive headsets. All three have made up 86% of the sales in Q3 2017 (Alto, 2017).

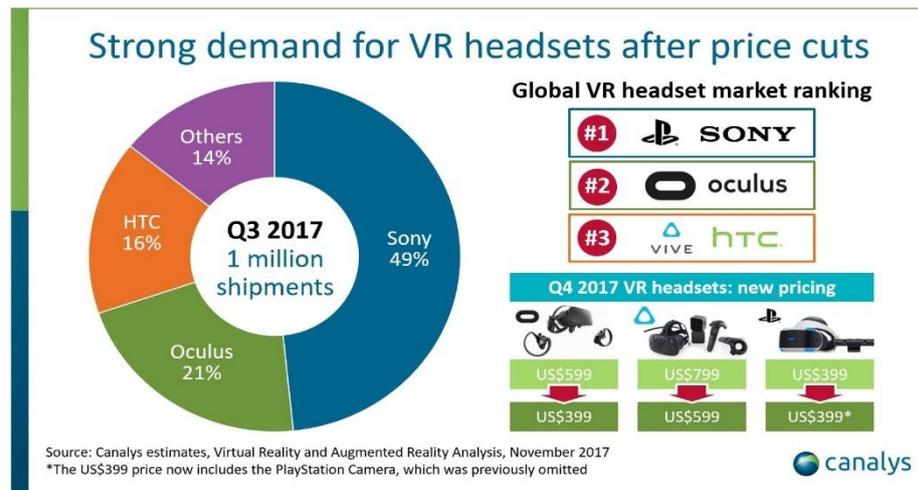


Figure 15 : Headset sales in Q3 2017 (Alto, 2017)

PlayStation’s lead was also due to its big market in Asia, holding more than 80% of the shares of the VR market in Japan. (Bradshaw, 2017). VR is popular in Japan due to their gaming culture which makes it an important market.

To appeal to more customers, HTC and Oculus will both be releasing a new mid-range autonomous headset respectively called Vive Focus and Oculus Go; both will be paired with a controller as well as an on board screen and processor. These will be standalone headsets and won’t require a computer to run it. Among these two, there are other headsets scheduled to release this year like the Pimax 8K, Lenovo Mirage Solo or Samsung Odyssey.

3.5 Types of interactions

This chapter will present the different types of interactions inside a virtual world.

3.5.1 Navigation

Movement in VR is very topical, as the body is static and our virtual player is moving, this may cause nausea to some users. There are different existing methods of navigation, some of which are more appealing to use than the other.

Using the **gaze**, the user will move in the direction they are looking. It is implemented mostly for VR using smartphones as they don’t have any input to change the direction. This system is usually found very uncomfortable by the user.

Using a **controller**, the user will move toward the desired direction when they want it by simply clicking a button or tilting the control stick. This system is more comfortable but the user is still static when movement occurs, so they can still suffer from motion sickness even with high-end headsets working with room-scale experience. Some experiences use the controller in other ways; released in February 2018, Sprint Vector is game where the user runs, flies and climb in a racing competition. Their *Fluid Locomotion* is done by swinging the controllers to run as if they were really running, tricking the brain that they are maybe really moving. This results in minimal or no discomfort at all (Sprint Vector, s.d.).

There are **other ways** to support movement. Some experiences rely on locomotion like driving which can be done seated where the user feels more comfortable. Some companies even created contraptions or tools like an omnidirectional treadmill where the user can walk indefinitely in the real and virtual world. Swiss company Somniacs SA created a full-body VR experience for flight simulation. The user wears the headset and lies down on their machine where they adopt a bird-like position; to move around, they will rotate their arms and flap their arms to regain altitude (_Birdly® - The Ultimate Dream of Flying, s.d.). As the user rises, falls or tilts in the virtual world, the machine will follow, and the body will feel the movement. A fan is placed in front of the user's head to simulate the wind. This experience being the author's most immersive VR experience ever tested.



Figure 16 : Author testing Birdly

The most common and preferred way of moving around the virtual world is the **teleportation**. The user passes from one point of view to the other, it can be determined points of view like for our application or it can be anywhere the user chooses. Some applications implement a blink effect as a transition when changing points of view to make it more comfortable. Our application will change the view when the user clicks on the appropriate button using their gaze. For many applications using a controller, a beam of light will emit from the virtual controller which allows the user to point in the position they want to be in

and when the movement is confirmed, the user will be immediately transferred to the new position.

3.5.2 Selection

This enables the user to start an action on a virtual object. It can be done by looking at it, by selecting with a device such as a controller or keyboard, by movement or by voice command. Selection can be followed by manipulation.

3.5.3 Manipulation

Manipulation allows the user to manipulate a 3D object and change its properties (position, rotation, size, etc.). In most 3D games, the player can hold different objects in the virtual world and mimic actions that they can do in the real world. For example, they see a baseball and baseball bat; with the controller in each hand, they can hold both objects, throw the ball in the air and take a swing with the bat. This is achieved using advanced equipment like the motion controllers, but other options exist.

One example is the Leap Motion, it is a sensor device that detects hand and finger motions. The Leap Motion was originally released in 2013 and now offers AR and VR support to high-end headsets and soon to mobile VR headsets (Reach into virtual reality with your bare hands, s.d.). It allows the user to interact with virtual objects with his bare hands and fingers.

Some wearable device also allows the user to track their body's motion. For example, Perception Neuron is a motion-tracking gear that ranges from single hand tracking to full-body tracking. The full-body tracking is composed of 32 captors which also includes the fingers. This allows the wearer to interact with his whole body. Perception Neuron can be used in Unity3D and Unreal Engine (Perception Neuron, s.d.).

3.6 Virtual Reality usage

This chapter will discuss the different industries that have used virtual reality.

3.6.1 Entertainment

The entertainment industry is the biggest producer of virtual reality content and is most often related to the gaming industry. Other industry has also found a use for virtual reality,

museum or exhibition centers offers interactive VR experiences to appeal to the younger generation who would normally see it as a lame experience. Some theme park offers virtual reality ride to offer an even more exciting experience; The Kraken Unleashed at Sea-World in Orlando, Florida, offers a VR experience aboard their roller coaster where the user is placed in submarine that is navigating through the danger of the deep sea while the cart is moving (Burt, 2018). A full virtual reality theme park has opened in February 2018 in Guizhou, China (Shamsian, 2017). There are more industries that have benefited from VR technology such as music, pornography, theaters...

3.6.2 Military

The military has had combat simulation for a long time, but with time, these were refined. These simulations are used by soldiers to improve their skills, for example a pilot can train with a flight simulator. VR isn't only used for vehicle simulation, it can transport the soldier in a specific virtual environment where they are faced with a certain scenario to which they must decide the next course of action. This trains the soldier to make decision in difficult circumstances. Battlefield simulation is another option where the soldier deals with a specific scenario (e.g. being under fire in a hostile environment) or where the commander sees a visualization of the battlefield and he must plan and evaluate the troop's possible movement.

Virtual reality is also used as a tool to treat veterans that suffer from PTSD; the idea is to trigger their trauma in a visualized situation that represents the traumatic event so the patient can slowly adjust and overcome it over time (Haar, 2005, pp. 1-2).

3.6.3 Healthcare

As for the treatment of PTSD in veteran soldiers, VR is used to help the patients to get rid of their fear or phobia such as the fear of heights or spider. It can also help patients with their physical therapy, doing exercises in a virtual game will be more entertaining than going to the gym. Virtual reality can go even deeper and help them with their cognitive rehabilitation, these patients, that suffer from illness such as a stroke, struggles with some basic everyday tasks. They can practice these tasks in a virtual environment and speed up the recovery process while still being supervised by medical staff (Powell, 2017). I

It is not only useful for the patient, doctors and surgeons can also benefit from this technology. Based on radiography, a virtualized version of the patient's anatomy that can be

used for diagnostic or for training purposes, the surgeon will be able to plan and practice the operation in VR before the actual operation (Mesko, s.d.).

During the Geneva Health Forum, the author demonstrated a few applications to showcase the potential of virtual reality in the health industry. Author noted that many people related to health sector, including big companies like La Croix Rouge, discussed about ongoing VR projects which shows that virtual reality is catching up and businesses are really seeing it as a valuable tool.

3.6.4 Education

VR offers many opportunities to teach a subject to students or even adults. The immersive nature of VR has been identified as a tool that facilitate the learning process (Tussyadiaha & Jia, 2017, pp. 1-2). The retailer Walmart is using virtual reality to train its employees, they are set in a virtual environment where they can prepare for a situation like dealing with rush crowds or cleaning up a mess made by a customer (Feloni, 2017). Google opened Expeditions VR, a tool that allows a teacher to virtually bring his students to different places like Antarctica or even a space station, and its AR counterpart where they can bring virtual elements to the classroom like a 3D version of the Earth.

Adventure-Lab created an augmented reality mat that offers educational content and games to the surface of the mat. With a smartphone or tablet, a child can see the virtual element on top of the mat and can learn more about the different area of a hospital or discover the different part of the human anatomy (Adventures-mat, s.d.). Adults can also learn using VR, we mentioned surgeon training on a virtual operation, but architects can also benefit from it; architects students can visualize in real time how their designs would look and decide if it works or not (Mercer, 2017).

During the Hacking Health Hackathon 2018, the author developed a VR simulation around medical interventions in extreme environments to train rescuers to act in difficult scenarios while still being in safety. The idea was offered by the Groupe d'Intervention Médicale en Montagne who needed a tool for them to train on realistic cases, hence the reason why they wanted a VR simulation. This again shows that VR is catching up.

3.6.5 Engineering

Some engineers use virtual reality as a part of the design process, this allows them to see their prototype in 3D to obtain a better understanding and detect flaws, quickly and easily

improving the overall quality of the concept. Elon Musk released a video in 2013 showcasing some prototype method of interaction with a complex 3D model of the SpaceX engine. Using a Leap Motion and an Oculus Rift, they were able to manipulate the engine and understand the fundamentals and how it should work (SpaceX, 2013).

Ford has a virtual reality lab where they create new vehicles using VR technology. Their lab, Ford Immersive Vehicle Environment Lab (FIVE), is a state-of-the-art facility where designers, engineers and researchers wear a VR headset to experience the concept of the car before manufacturing a physical prototype. They can evaluate the material, colors, aspect, ergonomics and more. This has improved production timing and reduced costs, resulting in Ford doubling their usage of FIVE since 2013 (Spears, 2017).

3.6.6 Tourism

VR is a strong marketing tool for the tourism industry, giving them the opportunity to attract more potential customers by offering a more compelling image of the destination, giving them a taste of what it is like to be there (Tussyadiaha & Jia, 2017). Consequently, helping them decide about where they want to travel and even enabling them to consider destinations they never thought about. The Wildlife Trust of South and West Wales was awarded £30,000 from Visit Wales to create 2 VR videos where the user can swim with dolphins off the coast of Pembrokeshire or where they can fly like a bird over the beautiful Welsh countryside (Gidley, 2017). Tourism Australia developed 18 different VR films to showcase many of the coastal and aquatic experiences that the country offers. Their initiative has led to an increase of 9% visitation and a 64% more engagement on Australia.com (Tourism Australia, s.d.).

Another advantage is that VR offers to the users an experience that they can't do or bring them to some places that they cannot go. North Korea is a complicated country to visit and many people are afraid to go there. SceneThere is a VR building tool that has created a virtual visit of the capital, Pyongyang. This allows anybody to explore different areas of North Korea and hear more information about the places and the country without physically going there. Their functionalities are similar to the ones in our application.

4 Development framework

In this chapter, we will analyze the different development frameworks that can be used for the creation of a virtual reality application.

4.1 Unity 3D

Unity 3D is a video game engine for 2D / 3D development and has been supporting virtual reality and augmented reality since December 2015. With Unity, it is possible to deploy over 28 platforms, including the main gaming platforms. Unity is free but also offers paid subscriptions (Plus and Pro) that offer additional features such as analytics, cloud storage, source code and more. Unity has a big community that provides, for free or at a cost, different assets or plugins in its marketplace, the *Unity Asset Store*.



Figure 17 : Unity's logo (Unity, s.d.)

Unity has been pushing its VR / AR development and is the most used platform for virtual reality development. Using the newest version (2017.3), Unity VR offers deployment for Oculus Rift, HTC Vive, PSVR, Android & IOS, HoloLens and Windows Mixed Reality (Unity for VR and AR, s.d.).

Unity supports C# as a programming language which is a very common scripting language.

4.2 Unreal Engine

Unreal Engine 4, made by Epic Games, is another notorious game engine that supports 2D / 3D / VR / AR development. Unreal supports the deployment of over 15 platforms, including the main gaming platforms. It is also free, but it does not have paid subscriptions. Its community is smaller than the one of Unity, so Unreal's marketplace offers fewer assets and plugins which also causes the assets to be more expensive.



Figure 18 : Unreal's logo (Unreal Engine, s.d.)

Using Unreal, it is possible to develop VR applications for Oculus Rift, HTC Vive, PSVR, Gear VR and Daydream.

Unreal supports C++ programming which is more performant and globally offers better graphics rendition.

4.3 CryEngine

CryEngine, made by Crytek, is the lesser-known game development engine. CryEngine only support development on Oculus Rift, Windows PC, Xbox One, PlayStation 4 and HTC Vive. It has a "Pay what you want " model which means that the engine and its source code is free, but any contribution is appreciated (Schreier, 2016). CryEngine has recently opened its marketplace but is still in the beta¹ phase causing the few assets available to be very expensive.



Figure 19 : CryEngine's logo (CryEngine, s.d.)

For virtual reality, CryEngine doesn't support mobile VR but supports development for Oculus Rift, HTC Vive and PSVR.

CryEngine supports C++ programming which is more performant. It is not the most popular game engine, but it has been used to create some of the games displaying the most advanced graphics at the time of release.

¹ An unfinished version of a program released for users to test before the final version is released.

4.4 Selected framework

The selected framework for the application is the Unity 3D game engine as the application will use the Google VR SDK. CryEngine does not support mobile VR and Unreal Engine only support GearVR and Daydream. Unity also offers better ways to implement 360° footage. Unity has an active community and a thorough documentation that we can use. If time and hardware weren't a constraint, the choice would have been Unreal Engine as C++ scripting is more efficient and the graphics of Unreal are more advanced.

5 Empirical part

This application is not issued for any commissioning party, it is a personal project. This project should be sought out so that the features developed, or the knowledge acquired during the process can be reused in another future VR project. Some details will be added to make the application more enjoyable to use and some feature will be implemented to prevent as much a possible the risk of nausea or any after effects. As the emergence of VR rose, more websites were developed to offer the consumers an easy way to create virtual reality websites or applications. This application will not use any online service and will focus on creating a proprietary solution. It can be achieved using the frameworks mentioned in the previous chapter (4).

5.1 Interactions

In the virtual world, the users should be able to interact inside the view they selected, providing them details about the view itself. The information will be communicated through two means:

1. A voice reading the text.
2. A pop-up window displaying the text.

After testing the prototype, the testers will report if the experience was immersive or not and which type of feedback was more immersive. This will be a factor when developing a future application.

For this application, different interest points between Helsinki and Switzerland will be recorded with a 360° camera and used as a virtual environment. In addition, the main menu will be set in a 3D virtual world to demonstrate to the user the difference between a virtual world and a 360° footage.

5.1.1 Types of interactions

The previous chapter defined the possible interactions and as the end goal is to make the application deployable with most compatible smartphones and headset, the users will use the gaze input module to click on the buttons; if their headset also contains a button, they will be able to use both methods.

Inside each view, the users will have 3 distinct buttons represented by a personalized icon:

The **movement** button allows to users to navigate to the next view. Before changing the view, the screen will fade to black and will fade back out to the next view.



Figure 20 : Movement button icon

The **audio** button will play a specific audio clip, audio files are handled so that the user can't click on the button multiple times. The file was made using a website that converts text to an audible speech (e.g. fromtexttospeech.com).



Figure 21 : Audio button icon

The **text** button will disappear when clicked, leaving at the same spot a pop-up window that will progressively type the text. The pop-up will include a close button, so the user can hide the panel when they finished reading the description. The original button will appear again when the pop-up is closed.



Figure 22 : Text button icon

5.2 Scenario

The application will be used with a headset. This application does not support positional tracking and so the user can only look around. When the user first launches the application and wears the headset, they will be able to look around. The Unity, HES-SO, Haaga-Helia logos will show up and load the next scene which introduces the concept of the gaze click input. Once they clicked on the button using the gaze input, they will be transported to the main menu which is set in a 3D environment. From here they can access the settings which include the choice to change the language to English or French.

In the menu, they have the choice between two paths, one for Finland and one for Switzerland. Each path has 3 scenes and each scene has one 360° footage as a virtual environment. Each scene is focused around one place which has something interesting to see; for Finland, the places are: the Sibelius monument, the Parliament, the Cathedral. The places were chosen because they are quite distinguishable and are often populated. For Switzerland, the places are: Hotel Chetzeron, HES-SO, Tourbillon's Castle. These places were chosen because the author frequently pass by them and they have something special to showcase.

Inside any of the six scenes, they are able to look around as long as they want, and they can click on the action buttons which will either display or play an audio clip to give them more information about the place, or it will transfer the user to the next scene. At all time, they can access the menu to update the settings or they can get back to the main menu to change the language or close the application.

Videos and background audio files will play on a loop as the user may spend a lot of time in a view. Videos are played on a loop because they only last for around 1 minute. During the recording, they were limited to 1 minute because a 4k 360° video takes a lot of time to export it to a format that Unity supports, and it also takes a lot of storage space on the phone.

This scenario is limited to this application, however, with the proper development, it can be extended to fit a larger scale application or answer a different need. Virtual reality is a great tool for discovery and learning, this application focused more on the discovery of different places. A future project with la Grande-Dixence is planned and will use the same features than this application, the project is to create a VR application that shows different places around the dam. It will be mostly used for tourists when the dam is closed or when it is too dangerous for tourists to walk around the dam.

Another possibility is to focus on the educational aspect, besides giving information to the user through text and audio, it is possible to create a tool for children to learn or train their knowledge of words in a foreign language. For example, if the user is playing a game to practice his German, they can be prompted to find "Die Katze" and will have to search the entire 360° view to find the cat. Oculus has a similar application called Gala360° which allows the user to discover the world through high quality 360° photos, the application also host educational content like a visit of the Oxford National History Museum. These concepts are not included in the research of the thesis.

5.3 Prototype development

Now that the core features of the application are defined, the following chapters will demonstrate how the features were created on Unity and how they were scripted. The Unity game engine will be used to create the application. Unity possesses a marketplace called Unity store which holds many different plugins or assets that can be downloaded (for free or upon transaction) and used in our project.

As a foundation for this project, the Unity documentation will often be the reference material and some of their tutorials as well as others found on different learning website will be followed to better understand the particularities of the framework. Unity is a popular game engine so if any trouble occurs during the process, it is most likely possible to find a solution or an alternative online.

5.3.1 Plugins used

This chapter will list the plugins used in the application.

5.3.1.1 Google VR (free)

Google VR provides SDKs for many popular development environments including Unity. This plugin offers to any developer the possibility to create VR application for Daydream or Cardboard. Some features covered by this plugin include (Quickstart for Google VR SDK for Unity with Android, s.d.):

- Spatial audio rendering
- Tracking user's head movement
- Input system for gaze, lateral button and controller
- Stereoscopic display
- Reticule representing the user's gaze
- Locomotion
- Object manipulation pointer
- Video player.

This plugin is essential for this application, it will access the gyroscope and the accelerometer of the phone. It also provides a simple way to test directly in the Unity window without having to build the application to a phone each time to test a feature, building on a phone generally takes two minutes or more and testing on Unity takes a few seconds.

5.3.1.2 GazeClick for Google VR (5\$)

This plugin has a simple feature, when the user looks at an interactive object such as a button, a small timer starts and when the time runs out, a click is simulated.

This will be the main method of interactions so with this plugin, the user has enough feedback to know when the click will occur. Without this, the user might click on a button by mistake when looking at it, but with the GazeClick, the user will need to focus a certain amount of time before the click occurs (Ateo, 2017). The default click with a button or finger is still usable, people using a Google Cardboard can either use this timed gaze function or simply press on the button of the headset to simulate the click faster.

5.3.1.3 ProBuilder (free)

This package is a plugin that enables us to quickly create and design level in the Unity editor. ProBuilder is a 3D modeling and level design tools, allowing to quickly prototype structures, complex terrain features. Probuilder has been used by many games. Previously a paid plugin, the development team has been hired by Unity Technologies and the asset is now free to download on the Unity Asset Store (Cimetiere, 2018)

This plugin will be used for the creation of a 3D virtual world that will act as the virtual world for the main menu.

5.3.1.4 Other plugins

The application will use other different plugins in the asset store.

- Unity Samples UI (free): This package allows us to quickly design a menu with some custom designed buttons and user interface instead of the basic Unity UI elements.
- Curved UI (25\$): This plugin is used to warp our canvas elements creating Curved UI similar to a curved TV. That way the user will feel as they are at the center of everything, increasing the level of immersion.
- IBM Watson (free): Watson is a powerful AI created by IBM. The Watson Unity SDK will be included to create a simple voice command tool.

Other websites will be used to download free assets like textures, 3D models, images, ...

5.3.2 Equipment

5.3.2.1 Camera

To capture our footage, a 360° camera model from Insta360, a company based in Shenzhen that creates affordable and accessible 360° camera, will be used. We will use their latest camera, the Insta360 one. This camera has many features (Insta360 One, s.d.):

- 24 Megapixels RAW photos.
- UHD 4K video.
- Time-lapse.
- Video stabilization.
- Bullet time.
- SmartTrack auto-framing.
- Live stream.

This is not the best performing camera on the market or even in Insta360's store but it offers the best compromise of price, quality and portability.

To make the best use of this camera, it will be mounted on top of a tripod to higher the position of the camera so that the point of view of the camera is at the same height as an average person's head.

5.3.2.2 Headset

The chosen headset is a custom one bought during the World VR Forum in Switzerland, but the application can be used with a cardboard or any other headset using a smartphone. In addition, the phone used is the Huawei P9.

5.4 Unity setup

This chapter will discuss the setup made for Unity. It will include how to prepare Unity for mobile VR development using the GoogleVR plugin and how to display a 360° video.

5.4.1 Creating and setting up a scene

In Unity, a scene contains the environment, game objects, menus and more. Think of each scene as a level. For the application, it will have a scene for each 360° footage along with other scenes for the introduction and menu. Unity possesses different type of game objects: 3D Objects, 2D Objects, UI, Audio, Light, etc. 3D Objects are predefined shapes

that can be modified, shapes varies from cube, sphere and more. These objects are physically rendered in a scene and this application will use both shapes mentioned. Another object that will be used often is an Empty GameObject, this object has no physical form and is used for organization or to add a script to the scene.

To create a new scene, we can go to *File > New Scene* and name it. Each scene comes with a default camera and a directional light. However, this camera does not yet provide a stereoscopic display. To enable the stereoscopic display and enable virtual reality development, it is necessary to import the Google VR plugin. To import a new package, it can be added directly when creating the project or, in this case, selecting *Assets > Import package > Custom package* and selecting the GoogleVR package that was downloaded from the Google VR website. Another way to import packages is from the Unity Asset Store window in the Unity editor. Once imported, the plugin will be located in the Assets folder; this Asset folder is the main folder for Unity development. Many subfolders will be created inside to organize the project.

Once the package is imported, there are a few components to add into the scene. The assets needed are in folder *GoogleVR / Prefabs*:

- GvrEventSystem.
- GvrEditorEmulator.
- GvrReticulePointer.

The last object needs to be a child of the Main Camera, for that, an empty game object is created using *GameObject > Create Empty* and renamed Player. Then, the Main Camera is moved inside the Player object (to be a child of the Player object) and the GvrReticulePointer is placed inside the Main Camera (to be a child of the Main Camera). To finalize the scene, the GazeClick package is imported through the Asset Store Window and the GazeClick object located in the GazeClick folder is added in the scene.

This will serve as a template for every future scene that will be implemented. When a new scene is needed, the template scene can be duplicated using the shortcut command Ctrl + d and should be renamed. Figure 23 shows the hierarchy of the template scene with the GoogleVR and GazeClick asset.



Figure 23 : Hierarchy of our template scene

The next step is to add the video player in the scene.

5.4.2 Video player

The GoogleVR plugin provides a way to display a 360° video but it didn't function as intended so another method was selected. The method to display a spherical video is to have the user placed in the center of a sphere and the video will be displayed inside the sphere.

The first step is to add an empty game object then, a sphere is added inside the empty object (*GameObject > 3D Object > Sphere*). When selecting the empty object, the inspector panel will open; this panel shows every component and configuration held by the object. While the empty object is selected, a Video Player component is added. The 360° video can already be added in the *Video Clip* field and the *Renderer* field can be filled with the sphere object. The video will render on top of the sphere, but the user will be placed inside the sphere, so they won't be able to view the video. To resolve that, the material of the sphere needs to render inside the sphere instead of outside (Unity, 2017). It can be changed using a custom material found online (VR 360 TV, 2017).

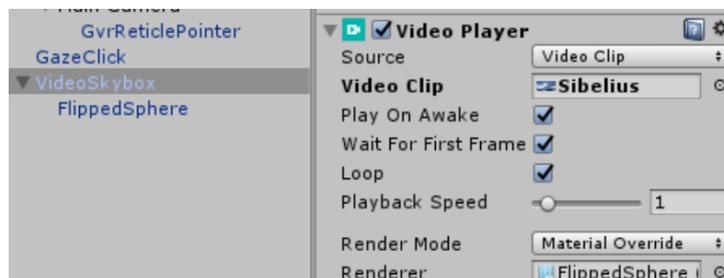


Figure 24 : Video player sphere

The users are now able to view a 360° video, it is important that the camera is in the center of the sphere so that it is at equidistance of every part of the video, an easy way to make sure of that is to set the position of all the components to the origin point (0 for all

axis). It is done that by selecting the object and clicking on the *gear icon* > *Reset* on the *Transform* option in the inspector. The sphere with the new material will look like the Figure 25.

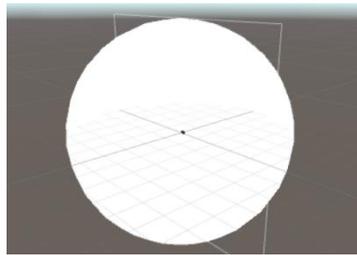


Figure 25 : Video player in a sphere

The quality of the audio output on the video is very bad; to resolve that, extra background audio was recorded. The video player component has an audio track field which receives an Audio Source object. An Audio Source is created (*GameObject* > *Audio* > *Audio Source*) as a sibling of the sphere and the audio file to play is selected. Not every audio format is supported in Unity, but it supports the common MP3.

Later, when adding the buttons to the scene, the sphere will be distracting. It can be easily disabled but it will be necessary to enable it each time we need to test the application. To make it easier, it is possible to replicate the effect with a script. The following script is attached to the sphere.

```
1. [ExecuteInEditMode]
2. public class HideInEditor: MonoBehaviour {
3.
4.     new Renderer renderer;
5.
6.     private void Awake()
7.     {
8.         //Gettig the renderer of the game object that is attached to the script
9.         renderer = gameObject.GetComponent < Renderer > ();
10.    }
11.
12.    void OnEnable()
13.    {
14.        //activate the renderer if not runing in editor mode or is in play mode
15.        renderer.enabled = !Application.isEditor || Application.isPlaying;
16.    }
17.
18.    void OnDisable()
19.    {
20.        renderer.enabled = true;
21.    }
22. }
```

The first step is to create a *Renderer* variable; the renderer is what renders the object and makes it visible. The first function **Awake()** is a lifecycle function called whenever an object is active in our scene, it is the first function called. In this function, the renderer of the

object is assigned to the local variable, the `gameObject` variable is the object to which the script is attached to. Through this variable, it is possible to call the function **GetComponent<>()**, this function will get the component in the object according to what is specified in between the tags.

The second function **OnEnable()** is called whenever the script is active; it decides to render the sphere if it is not running inside the editor mode or if it is playing. The third function **OnDisable()** is called whenever the script is deactivated, this will always render the sphere.

5.4.3 Building the application to smartphone

To be able to test the application on a mobile phone, there are a few steps to follow beforehand. First, the development platform in *File > Build Settings* needs to be switched to Android. If no Android SDK is found, Unity will redirect to the Android page to download the SDK. Once the platform is switched, the *Build System* has to be changed from Gradle to Internal. Finally, when clicking on *Player Setting*, this will open a new panel in the Inspector; at the bottom is located the *XR Setting* where it is necessary to make sure that the *Virtual Reality Supported* checkbox is ticked and that the *Virtual Reality SDKs* is Cardboard. Before building the application, the scenes to play must be loaded in the Build Setting. To load them, the scenes can be dragged into the Scenes In Build field.

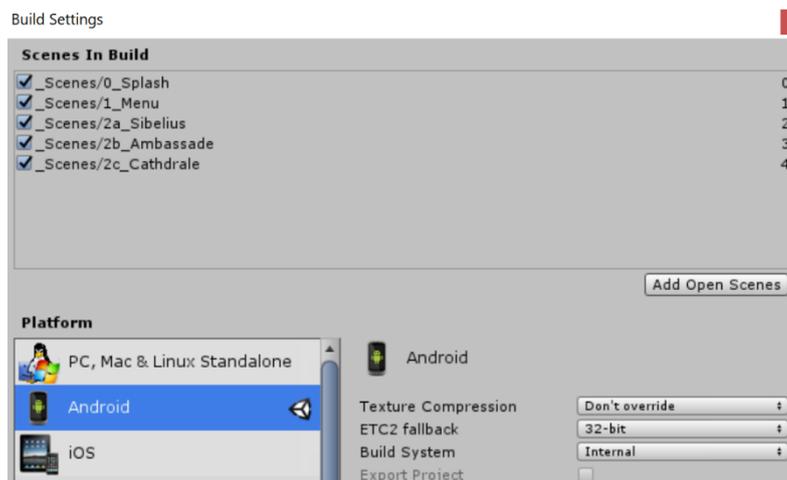


Figure 26 : Build settings

The configuration for Unity is now done, now we just need to configure the phone.

Like for most mobile focused development, the phone must have the developer options enabled. To do so, we need to go to *Settings > About phone > Build number* and click on the Build number 7 times; this will unlock the developer options which are accessible in

the Settings (Stimac, 2014). There, *USB Debugging* needs to be activated. The application can now be exported to the smartphone.

5.5 Unity development

This chapter will discuss the different features implemented in the application.

5.5.1 Blink effect

For the blink effect, we use a cube that will be covering the main camera as if the user has a box on his head hiding his vision. We create a cube (*GameObject > 3D Object > Cube*) and a new material (*Assets > Create > Material*). We need to change some properties of the material; first, the Albedo color has to be set to black (`#FFFFFF`) and the Rendering Mode must be changed to Fade. We can then change the default material of the cube to the one created and set the cube as a child of the Main Camera. The cube needs to be a child of the main camera so that when the user moves, the cube will follow and continue to hide their field of view.

A script will manipulate the alpha of the cube which is similar to controlling the transparency. The alpha value ranges between 0 and 1; 0 being transparent and 1 being opaque. The cube will fade out when the scene is loaded and will fade back in when the corresponding method is called. The alpha value can be found in the Mesh Renderer of the cube.

In the script, a local variable will hold the value of the Renderer object as it will be accessed multiple time throughout the script. With direct access to the cube's renderer, it can start fading out the cube one second after the scene has loaded to make sure that the video is loaded properly as 360° 4K videos are heavy to load. The cube will fade back in just before changing a level, level management will all be handled in a separate script.

```
1. public class FadeIn: MonoBehaviour {
2.
3.     public float fadeInTime;
4.     private Renderer renderer;
5.     private Color currentColour = Color.black;
6.     static int fadeDirection;
7.
8.     void Start()
9.     {
10.        //getting the Mesh Renderer of the cube attached to the script
11.        renderer = gameObject.GetComponent < Renderer > ();
12.        //Start fading animation
13.        FadeOut();
14.    }
15.
```

```

16.     void Update()
17.     {
18.         //waiting one second for the video to load appropriately
19.         if (Time.timeSinceLevelLoad > 1) {
20.             //execute the code only during transition
21.             if ((fadeDirection == 1 && currentColour.a < 1) || (fadeDirection == -
22.                 1 && currentColour.a > 0)) {
23.                 currentColour.a += fadeDirection * fadeInTime * Time.deltaTime;
24.                 currentColour.a = Mathf.Clamp01(currentColour.a);
25.                 renderer.material.color = currentColour;
26.             }
27.         }
28.     }
29.
30.     public static void FadeBack() {
31.         fadeDirection = 1;
32.     }
33.
34.     void FadeOut() {
35.         fadeDirection = -1;
36.     }

```

Regarding the first variables, fadeInTime is used to define how long the fading animation should last. By setting this variable to public, its value can be updated in the Unity editor, in the script component of the object.

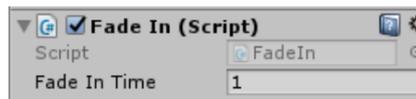


Figure 27 : Public variable in editor

Objects can also be declared as public, we will use this functionality later in other scripts. The color variable is simply a RGBA color object. It is declared as black, so only the alpha value will be updated. The fadeDirection variable will define if the alpha value will decrease or increase. It will switch between two values: 1 and -1.

In the **Start()** function, we get the renderer of the cube and the function to fade the cube out is called. The start function is another lifecycle functions that occurs when the object is active but is called after the **Awake()** function.

The **Update()** function is the most commonly used functions, it is called each frame after the **Start()** function. In this function, we first check if the time since the level has loaded is over one second, this should leave enough time for our video to render properly on our sphere. Then, to avoid manipulating the alpha value each frame, we write a conditional statement that will check if the fading is finished, if not, it will decrease or increase the alpha according to the fade direction. The Time.deltaTime is a value used by many developers in graphics programming. It represents the time needed to render the last frame. In our application, it ranges from 0.015 to 0.025 second when fading out the cube. If the fading out animation last one second, then our alpha will decrease by 0.015 (base value is 1)

each frame until it reaches zero. The **FadeBack()** function is declared as static because we want to access it from within another script, because of that, the `fadeDirection` variable need to be declared as static as well.

5.5.2 Level manager

To handle level management, we add an empty game object to the scene and add a script to the object. We can easily handle the change using the `SceneManager` class. This class has a method called **LoadScene()**, taking either a string (name of the level) or an integer (the index of the scene according to the build settings). Loading a level is done instantly but we do not want to jump straight from the video to the black screen (it is black because of the cube in the next scene). We implemented a method to fade the cube back, but it does not have the time to start the fade in animation as the level has already changed. It can be resolved by setting a timer to wait one second for the animation to complete. The technique to achieve this is to implement a coroutine that will wait one second before calling the function.

```
1. public class LevelManager: MonoBehaviour {
2.
3.     public float autoLoadNextLevel;
4.
5.     private void Start()
6.     {
7.         if (autoLoadNextLevel == 0) {
8.             Debug.Log("auto load disabled");
9.         } else {
10.            //invoke method can not take parameter
11.            Invoke("LoadNextLevel", autoLoadNextLevel);
12.        }
13.    }
14.
15.    public void LoadNextLevel(string name)
16.    {
17.        StartCoroutine(LoadLevelWithDelay(name));
18.    }
19.    //Coroutine
20.    IEnumerator LoadLevelWithDelay(string name) {
21.        //calling static method to fade back the cube
22.        FadeIn.FadBack();
23.        //We wait 1 seconds for our blink effect to execute
24.        yield return new WaitForSeconds(1);
25.
26.        //Load scene according to name
27.        SceneManager.LoadScene(name);
28.    }
29.
30.    public void LoadNextLevel() {
31.        //load following scene according to build setting index
32.        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
33.    }
34.
35.    public void CloseApplication() {
36.        Application.Quit();
37.    }
38. }
```

The first **Start()** function is used if we want to automatically load the next scene in the build index, it is useful for a splash screen. The second function **LoadNextLevel(string name)** require a string parameter, the string will be the name of the scene. This function will call the coroutine function **LoadLevelWithDelay(string name)** and pass the string parameter to the coroutine. The coroutine calls the static method defined in the previous script to fade the cube back and will wait for one second before executing the next piece of code which is loading the next scene using the SceneManager class. The **LoadNextLevel()** function loads the next level by searching the index of the current level and increasing it by 1. This function is used in the Start function. The last function will close the application.

5.5.3 Interactive button

This will be the main method of interaction inside a scene. To create a button, we select *Game Object > UI > Button*. All the UI interface should be inside a canvas object that is created along with the 1st UI element. By default, canvas has a *Render Mode* of Screen Space – Overlay, which lays the UI elements on top of the camera; this mode is not useful as the button will be placed somewhere in the world. Conveniently, the canvas render mode has three options which include World Space; with this option, the button can be placed freely in the scene and will behave as an object.

The button will be composed of one image and one text. For the image, a few images were downloaded on FlatIcon and modified to suit the application. Once the icons are ready, they can be imported in Unity. When importing images that will be used in a menu, the *Texture Type* has to be changed to Sprite (2D and UI). To add an image, we first need to remove the *Image(Script)* from the button as we have our own image. We can then add an image (*GameObject > UI > Image*) as a child of the button. The custom image can be selected in the *Source Image* field. The image will probably appear too big in the virtual world and will require to be scaled down.

Once the button is ready and adjusted, we start working on its animation. We want the text to be shown when our gaze is hovering the button and to remain hidden when looking elsewhere. Unity has an entire animation system based on state machine hierarchy. To animate the button, an Animator component is added. The animator requires a *Controller* which is created selecting *Assets > Create > Animator Controller* and assigned to the animator. We will not dive too deep into how animation works as it takes a lot of times. Figure 28 shows the animator's states.

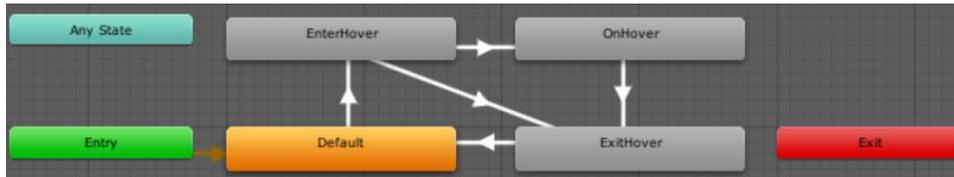


Figure 28 : Our button's animator state

The animator has 4 different states:

- Default : Our icon displayed without the text.
- EnterHover : The text slowly moves and appears.
- OnHover : Both icon and text are displayed.
- ExitHover : Our text moves and disappears.

To animate an object, each of our state can be edited in the *Animation* window. In this window are located a timeline and a property windows. We can access the properties of the object containing the controller as well as its child; in our case, the button has the controller, so we can access the image and the text. For the animation, we will manipulate the position and the color of the text over a defined period of time. The animations will last less than 1 second.

Figure 29 shows the animation timeline of the EnterHover at the end of the animation.

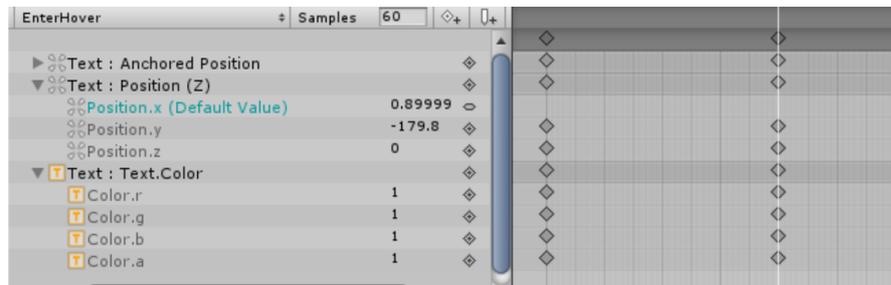


Figure 29 : EnterHover animation panel

The controller can hold parameters of different types (Float, Int, Boolean, Trigger), we create a Boolean parameter which will change when the cursor is hovering on the button. This parameter will define when a state transfers onto another state. The controller's parameters can be accessed and updated via a script.

To call a function when hovering the button, it will need an *Event Trigger (Script)*. There are different types of triggers, but the ones implemented for these buttons are the *Pointer Enter* and *Pointer Exit* which are triggered when the gaze is looking at a button or when it

has stopped looking at it. When adding a trigger, it will have an object and a function field to call a function from the selected object. In this case, a new script is added to the button.

```
1. public class ButtonController: MonoBehaviour {
2.
3.     Animator animator;
4.
5.     void Start()
6.     {
7.         //getting the Animator component
8.         animator = GetComponent < Animator > ();
9.     }
10.
11.     public void SetHoverTrue()
12.     {
13.         animator.SetBool("HoverEnter", true);
14.     }
15.
16.     public void SetHoverFalse()
17.     {
18.         animator.SetBool("HoverEnter", false);
19.     }
20. }
```

The **Start()** function will assign the Animator component to the local Animator variable. Finally, we create two functions to switch the value of the animation's parameter to true or false. The SetBool method receives two parameters, the first one is the name of the parameter defined in the animator and the second is the new value of the parameter. The functions can now be called on their corresponding trigger by adding the button to the function field in the triggers.



Figure 30 : Animated button

Figure 30 shows the first state of the button when the user is not looking at it, the second image show the button and its corresponding text when the user is looking at the button. Finally, the third image shows the text which is moving and disappearing when the user looks away from the button. These functions will be called in the appropriate trigger.

The buttons will be placed freely in the virtual world, but it should be visible as if it is in front of the user. This is hard to replicate by setting it up manually as the button must be angled at the right degree. Unity has a function to force an object to look at a specific

other object. In our case, the button will look at the player / camera. The following code shows how to force the button to face the player.

```
1. public class LookAtPlayer: MonoBehaviour {
2.
3.     void Update()
4.     {
5.         transform.LookAt(2 * gameObject.transform.position - Vector3.zero);
6.     }
7. }
```

This function will determine the direction to look at by calculating the difference between the position of the object and the position of the camera which we have defined as Vector3.zero because the camera is set at the origin point (0,0,0 on all the position's axis).

This will be the base for all the buttons, the only things to change will be the icon and the text. The last step is to define the action of the button, a button has an *On Click ()* field where multiple actions can be defined.

5.5.3.1 Navigation button

A button can handle multiple function during a click event. For the first button, the navigation button, we add a new On Click entry and drag the Level Manager to the field, then we need to select the appropriate function. In the function call field, we have a list of all the component function in the object, we need to select the LevelManager script and select the LoadNextLevel function. By doing so, a new field will appear because the function receives a string parameter. In this field, the name of the next scene is specified.

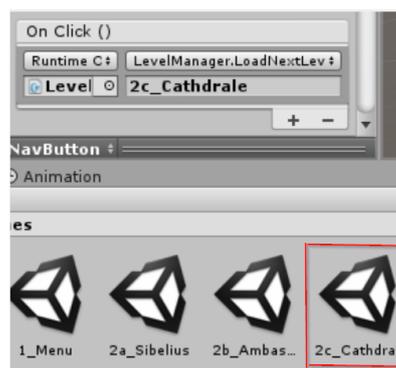


Figure 31 : On Click example

5.5.3.2 Audio button

To handle the audio interaction, we create an empty game object and attach our Audio Manager script.

```

1. public class AudioManager: MonoBehaviour {
2.
3.     private AudioSource audio;
4.
5.     public void PlayAudio(AudioSource audioToPlay)
6.     {
7.         //if no audio, we assign it
8.         if (audio == null) {
9.             audio = audioToPlay;
10.        }
11.
12.        //check if no clip is playing to play the new audio
13.        if (!audio.isPlaying) {
14.            audio = audioToPlay;
15.            audio.Play();
16.        }
17.    }
18. }

```

In case there are multiple audio files to play, we want to make sure that the audios don't overlap. Our local `AudioSource` variable will hold the audio file to play. The **PlayAudio(AudioSource audioToPlay)** function takes an audio source object as a parameter, it will assign the parameter to the local variable if the audio is empty at first. Then, it checks if the audio clip is already playing; if yes, nothing will happen, otherwise, it will assign the parameter to the local variable again in case the file has changed and the file is played.

In the scene, we will add an `AudioSource` object as a child of the button. In the inspector, the object has an `AudioClip` field for the audio file to play, it also has a checkbox *Play on Awake* which will play the audio when it is first initialized. This option is unchecked as it must only play when the user decides it.

The component is ready now, so we need to handle the click event. In the `On Click` field of the button, we add a new entry, select the `Audio Manager` and its `PlayAudio` function, then drag the `Audio Source` object as a parameter.



Figure 32 : Audio button On Click example

5.5.3.3 Text button

In this interaction, the button will be replaced with a panel whenever it is clicked and will only appear again once the panel is closed. In the same canvas as the button, we add a panel object (*GameObject > UI > Panel*). Inside the panel, we add a text object that will hold the information and a button to close the panel. After adjusting the size and position of the text, button and panel, we need to deactivate the panel as it shouldn't be visible at

first. Deactivating an object is done by selecting the object and unchecking the box at the top of the inspector.



Figure 33 : Example of button and panel overlaying

To switch between the button and panel, the appropriate functions need to be called on the two buttons. Our custom button needs 2 functions, the first is to hide the button itself and the second to display the panel. This can be done by adding a On Click entries and adding the object to hide / display in the field; we can then access the SetActive function inside the GameObject class and define its value.

Figure 34 shows the functions called when clicking the button

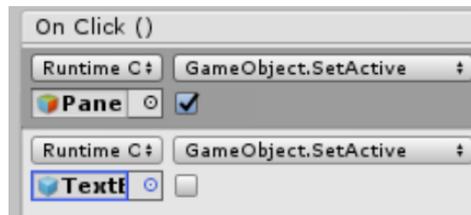


Figure 34 : Text button On Click example

All the buttons are ready, we can create Prefabs to reuse the button. To create a prefab, we can simply select the object in the hierarchy and drag it into the asset folder. This way, when another button is needed, it can be quickly replicated using the prefab.

5.5.4 Menu

The menu will be the second view the user will see, they will be able to:

- Select the tour (Switzerland /Finland)
- Get information about the application
- Access the settings menu to
 - Change volume of audio file
 - Change volume of video
 - Change the language (only in main menu)
- Exit the application.

The menu will function similarly to JAVA's CardLayout which means that the canvas will hold many panels overlapping each other but it will only display one at the time or none at all. To create the first panel, we must first add a new canvas, change its *Render Mode* to World space, we can then add a new panel to the canvas and resize it. For the panels, buttons and other UI elements, we are using the variations that comes with the Unity Samples UI package, but it can be done without as it is purely an aesthetic addition.

5.5.4.1 Panel creation

The first panel will be the main menu panel, it will hold 4 buttons: Start, About, Settings and Close. To lay the buttons with the same spacing and padding between them, it is possible to add a Vertical Layout Group component which will position its child elements on top of each other. The vertical Layout component has parameters to define the spacing between the elements, the padding for each side of the elements and the alignment of its child. After adding the first button as a child of the panel, the button can be duplicated three times and they will all be properly aligned. Figure 35 shows the first panel with its buttons along with the configuration of the vertical group component.

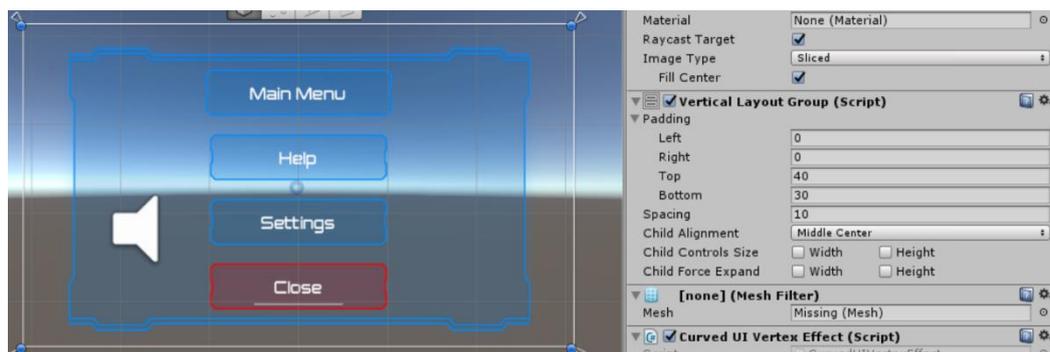


Figure 35 : Main menu panel with vertical group component

The second panel will be a simple panel holding one button for each country. To align them one next to the other, a Horizontal Layout group was added to the panel. Figure 36 shows the two options on the panel.

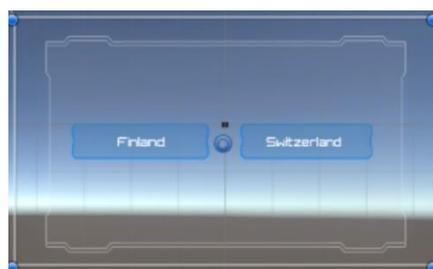


Figure 36 : Country selection

The third panel will appear when clicking the about button. This panel will hold another panel containing a text and a button to navigate back to the Main menu panel. As it doesn't need the same space between the elements, no Vertical Layout Group is needed. Figure 37 shows the About panel.

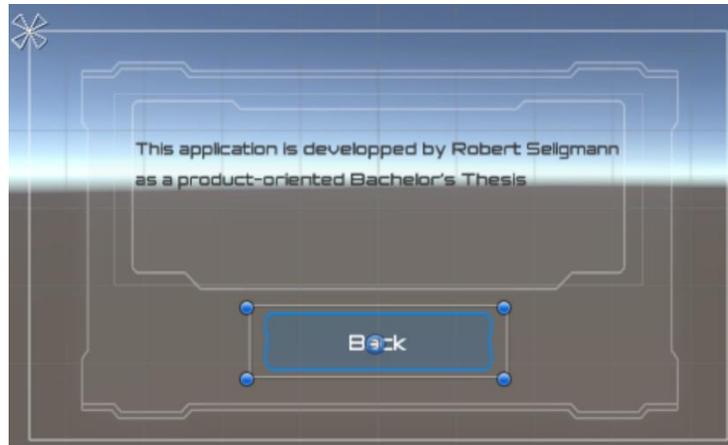


Figure 37 : About panel

The fourth panel is the settings panel, the volume for the videos and audios can be changed there as well as the application's language. When changing the values, these changes need to be persistent even if the user closes the application. Unity has a PlayerPrefs class which stores player preferences on the device and accesses it during sessions. It will be necessary to add or change the script on both the video player and the audio button to check the player preferences and set its volume. The data is stored as a Key / Value pair. The key is a string name defined by the developer and the value is a one of the predefined variable types.

```
1. public class PlayerPrefsManager1: MonoBehaviour {
2.
3.     const string MASTER_BACKGROUND_VOLUME = "background_volume";
4.     const string MASTER_VOICE_VOLUME = "voice_volume";
5.     const string MASTER_LANGUAGE = "language";
6.
7.     public static void SetMasterBackgroundVolume(float volume)
8.     {
9.         if (volume >= 0 f && volume <= 1 f) {
10.             PlayerPrefs.SetFloat(MASTER_BACKGROUND_VOLUME, volume);
11.         } else {
12.             Debug.LogError("Background Volume out of range");
13.         }
14.     }
15.
16.     public static void SetMasterVoiceVolume(float volume)
17.     {
18.         if (volume >= 0 f && volume <= 1 f) {
19.             PlayerPrefs.SetFloat(MASTER_VOICE_VOLUME, volume);
20.         } else {
21.             Debug.LogError("Voice Volume out of range");
22.         }
23.     }
24. }
```

```

25.     public static float GetMasterBackgroundVolume()
26.     {
27.         if (!PlayerPrefs.HasKey(MASTER_BACKGROUND_VOLUME)) {
28.             SetMasterBackgroundVolume(0.5 f);
29.         }
30.         return PlayerPrefs.GetFloat(MASTER_BACKGROUND_VOLUME);
31.     }
32.
33.     public static float GetMasterVoiceVolume()
34.     {
35.         if (!PlayerPrefs.HasKey(MASTER_VOICE_VOLUME)) {
36.             SetMasterVoiceVolume(0.5 f);
37.         }
38.         return PlayerPrefs.GetFloat(MASTER_VOICE_VOLUME);
39.     }
40.
41.     public static void SetLanguageToFrench()
42.     {
43.         PlayerPrefs.SetString(MASTER_LANGUAGE, "FR");
44.     }
45.
46.     public static void SetLanguageToEnglish()
47.     {
48.         PlayerPrefs.SetString(MASTER_LANGUAGE, "EN");
49.     }
50.
51.     public static string GetSelectedLanguage()
52.     {
53.         if (!PlayerPrefs.HasKey(MASTER_LANGUAGE)) {
54.             SetLanguageToEnglish();
55.         }
56.         return PlayerPrefs.GetString(MASTER_LANGUAGE);
57.     }
58. }

```

The three variables are constant string that represents the key value, creating a constant variable is a common practice to avoid mistyping the key throughout the code. The volume of a video or audio file ranges from 0 to 1, so their key will be paired with a float value and the language will be a string value. The function **SetMasterBackgroundVolume(float volume)** takes the new value of the volume for the video and checks if the value is within the valid range and saves the new key / float value in the PlayerPrefs if the value is correct. The second function **SetMasterVoiceVolume(float volume)** works on a similar scheme except that it defines the volume for the audio files.

The two next functions will check the PlayerPrefs for the volume keys and return its value; if the key doesn't exist, which is the case when the application is first deployed, the script will save the value of 0.5 for both volume and return the value.

For the language, the application will support only English and French. Two functions are created to store their respective string value. The **GetSelectedLanguage()** will first check if a value is stored and will set the default value to English if nothing is found, otherwise, it will return the stored value.

All the functions are static so that they can be called from another script without creating an instance.

The script for the settings is done, the next step is the settings panel. The first iteration of the panel included two sliders representing the value of the volume. Unfortunately, VR technology is still a process in development, causing the slider to behave in an unexpected manner when trying to change its value using the gaze input. The slider functions well when the user clicks the screen manually with the button on the headset, but the application needs to be streamlined so it can be usable with every mobile headset. To work with our gaze input module, we have adapted the components and layout of the panel. The sliders are now only indicative, the *Interactable* option on the slider has been deactivated and cannot be clicked anymore; to change the volume, an alignment of ten buttons is placed under the slider to change the volume. The buttons are the children of an empty game object holding a Horizontal Layout Group component to properly align them one next to the other. Figure 38 shows the layout of the setting panel and the group of buttons.

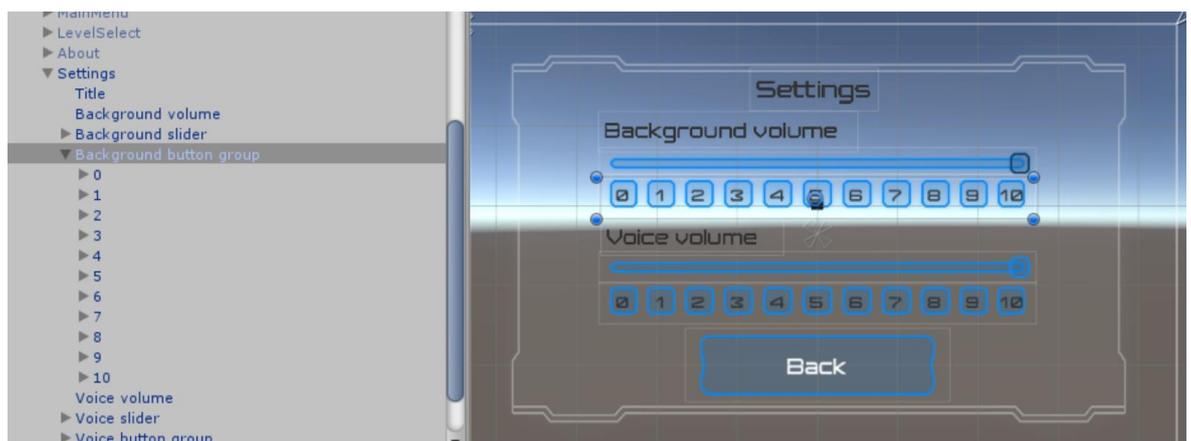


Figure 38 : Layout of the setting panel

Clicking on one of the ten buttons will change the value of the slider and when that change occurs, the player preferences must be updated. The On Click event of the button will hold the slider and call its value function to change the value to the one passed into the parameter field. In the Main Menu, the settings panel will also have two buttons to call the functions to change the language. To handle any updates on the slider, an empty game object is added to the scene and a new script is attached to it.

```
1. public class OptionsManager: MonoBehaviour {
2.
3.     public Slider backgroundSlider;
4.     public Slider voiceSlider;
5.
6.     void Start()
7.     {
8.         backgroundSlider.value = PlayerPrefsManager1.GetMasterBackgroundVolume();
9.         voiceSlider.value = PlayerPrefsManager1.GetMasterVoiceVolume();
```

```

10.     }
11.
12.     void Update()
13.     {
14.         //Calling our PlayerPrefsManager class and setting the volume
15.         PlayerPrefsManager1.SetMasterBackgroundVolume(backgroundSlider.value);
16.         PlayerPrefsManager1.SetMasterVoiceVolume(voiceSlider.value);
17.     }
18.
19.     public void ChangeLanguageToFR()
20.     {
21.         PlayerPrefsManager1.SetLanguageToFrench();
22.     }
23.
24.     public void ChangeLanguageToEN()
25.     {
26.         PlayerPrefsManager1.SetLanguageToEnglish();
27.     }
28. }

```

In the script, two Slider variables are declared as public, so we can define them with the one we created, the value of the slider can be accessed through those variables.

In the **Start()** function, the static functions created in our PlayerPrefsManager will return the value saved in the player preferences and assign it to the value of the slider. The **Update()** function is called each frame and will save the value of the slider to the player preferences. Finally, two methods are included to change the language of the application, these methods can be attached to a button's On Click event.

The changes made for the volume need to update the volume of the audio file. To do so, another function needs to be added to the AudioManager.cs script.

```

1. void Update()
2. {
3.     if (audio != null) {
4.         audio.volume = PlayerPrefsManager1.GetMasterVoiceVolume();
5.     }
6. }

```

The volume can be accessed directly in the audio variable, calling the static method **getMasterVoiceVolume()** will return the value saved in the player preferences.

For the volume of the video, a script is added to the Audio Source located in the sphere.

```

1. public class VideoAudioPlayer: MonoBehaviour {
2.
3.     AudioSource audio;
4.
5.     void Start()
6.     {
7.         audio = GetComponent < AudioSource > ();
8.     }
9.
10.    void Update()

```

```

11.     {
12.         audio.volume = PlayerPrefsManager1.GetMasterBackgroundVolume();
13.     }
14. }

```

The **Start()** function will get the Audio Source attached to the script and assign it to the local variable. The **Update()** function is called each frame and will assign the volume saved in player preferences to the volume of the audio file.

The settings panel is complete, the last button in the Main menu is the Exit button which calls the **CloseApplication()** function in the LevelManager object.

5.5.4.2 Panel switch

All the panels are finished but they are still overlapping, the canvas holding the panel require a script to handle the display of the panels.

```

1. public class MenuContoller: MonoBehaviour {
2.
3.     public List < GameObject > menus; //List of every menu panel
4.     public int menuSelected = -1; //the menu to display according to list, set to -
    1 to display none
5.
6.     public void SwitchMenu(int selected)
7.     {
8.         //setting new value
9.         menuSelected = selected;
10.    }
11.
12.    private void RenderMenu()
13.    {
14.        //loop the list and display the selected one
15.        for (int i = 0; i < menus.Count; i++) {
16.            if (i == menuSelected) {
17.                menus[menuSelected].SetActive(true);
18.            } else {
19.                menus[i].SetActive(false);
20.            }
21.        }
22.    }
23.
24.    void Update()
25.    {
26.        RenderMenu();
27.    }
28.
29.    public void HideUI()
30.    {
31.        // set the value to negative to hide all menus
32.        menuSelected = -1;
33.    }
34. }

```

This script has two variables. The first one is a list of GameObject, in this case the game object will be a panel. By declaring the list as public, the inspector has an editable field to define the size of the list and will create other fields depending on the size of the list. The

main menu has 4 panel, so when defining the size of the list to 4, the inspector will have 4 new fields to hold 4 GameObjects. The second variable is the index of the list to display, by default it is set to -1 so it doesn't show any panel. The *Menu Selected* field defines the panel to display according the list, it is important to note that C# lists start at 0 (zero). Figure 39 shows the public variables in the inspector.

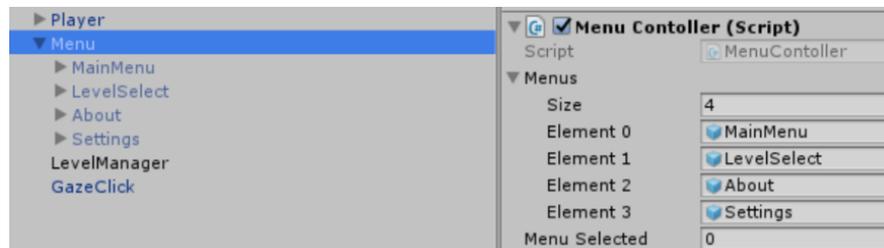


Figure 39 : Menu Controller public variables

The function **SwitchMenu(int selected)** change the value of the local variable. This function will be mostly called when clicking on a button to change the panel to display. The **RenderMenu()** function loops in the list of panels and checks if the index of the object is equal to the one to display. If the test returns true then the panel is set to be active otherwise, the panel is set to be inactive. Finally, the **Update()** function renders the menu each frame and the **HideGUI()** function change the selected menu variable to -1 so that no panels are displayed.

Switching between panel is easily done now by calling the SwitchMenu function in the canvas when clicking on a button. Figure 40 shows an example of an On Click event to switch the panel.



Figure 40 : Menu switch On Click example

The main menu is now finished. The next step is to make it accessible in any scene, so the user can get back to the main menu or access the settings.

5.5.4.3 Menu pop-up

Many VR smartphone experiences display the menu when the user is looking down at his feet, we will replicate this action in this application.

First, an empty GameObject is added in the scene as a child of the Player object. Reminder: in the scene, the Player object is an empty game object holding the Main Camera. A panel is then added to the empty object and positioned under the player as if it is the ground.

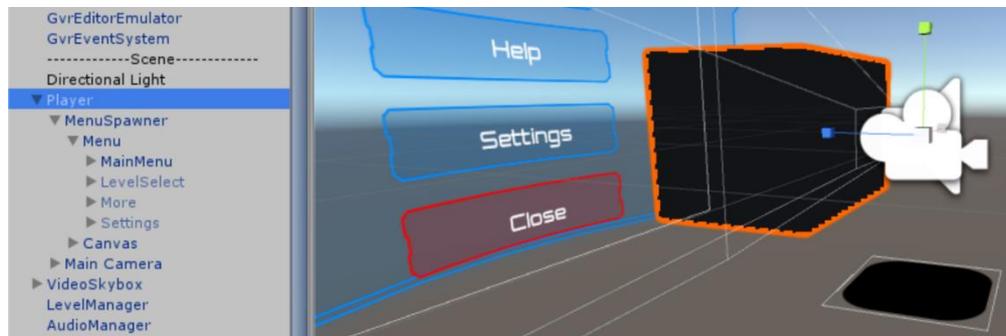


Figure 41 : Player object with the panel

Afterwards, the menu created will be copied and pasted as a child of another empty object, this empty object is set as a child of the Player object as well. The menu is placed in front of the user. This version of the menu is slightly different from the main one as this menu is only seen when they are visiting a location; the Exit button will be replaced with a Close button to hide the menu, the start button will be replaced with a main menu button and the menu selected value is set to -1 so that the menu is hidden at first, the language buttons in main settings are not included in this version as well. A script is added to the empty object to make the menu appear.

```

1. public class MenuAppear: MonoBehaviour {
2.
3.     private Camera camera; //Main player camera
4.     private MenuContoller menuContoller;
5.
6.     void Start()
7.     {
8.         //Getting the menu script holding all menu panel
9.         camera = Camera.main;
10.        menuContoller = gameObject.GetComponentInChildren < MenuContoller > (true);
11.    }
12.
13.    public void DisplayMenu()
14.    {
15.        //Turning the object at the same degree than the camera
16.        gameObject.transform.eulerAngles = new Vector3(0, camera.transform.eulerAngles.y, 0); //display the first menu
17.        menuContoller.SwitchMenu(0);
18.    }
19. }

```

The script holds a Camera and a MenuContoller variable. The camera variable is required because the menu needs to appear in front of the user. The menu is a child of the empty object so when the object rotates, the menu will move along the object. Figure 42 explains the phenomena, in this example the cylinder is the empty object.

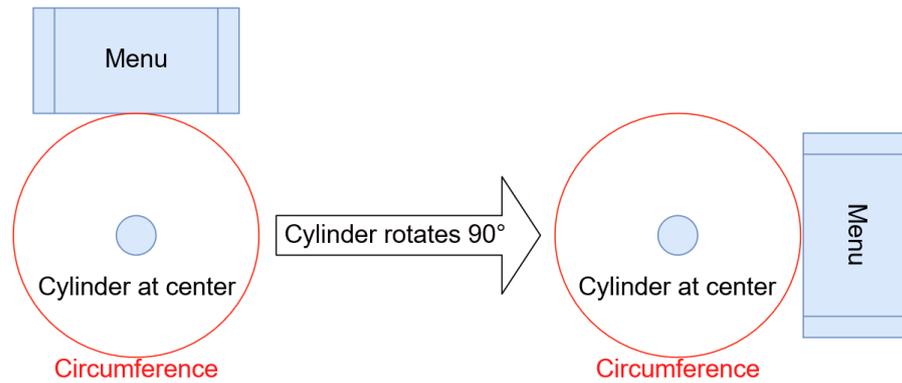


Figure 42 : The rotation of the cylinder

The camera variable is used to turn the empty object at the right degree. The menuController variable is accessed to call the **SwitchMenu()** function. In the **Start()** function, we assign the objects to the variable. The Camera.main function return the first main camera, as the scene has only one camera, it is not a problem. **The GetComponentInChildren<MenuContoller>(true)** searches the children for the MenuController script, the parameter true means it will search even the objects that are deactivated.

The **DisplayMenu()** will turn the empty object so the menu will be in front of the player. A Vector3 is a 3-dimensional vector which represents the x, y and z axis, the rotation needed in this case is the y axis. When the user rotates 90°, a new Vector3(0, 90, 0) is assigned as the value of the empty object's position. Once the menu is rotated in front of the user, the script calls the **SwitchMenu()** function to display the menu. The **DisplayMenu()** function is called in a *Pointer Enter Event Trigger* on the panel.

Figure 43 shows what happens when the user looks down and up.



Figure 43 : Menu Pop-up

5.5.5 Multi-language

5.5.5.1 Text support

This application will support the English and French language but can be extended to other languages. The language to display will depend on the value stored in the Player-Prefs. To support both languages, a new script is created and will be added to every text component.

```
1. public class Multi_Language_text: MonoBehaviour {
2.
3.     public string french;
4.     public string english;
5.     public bool progressive;
6.     private Text description;
7.
8.     void Awake()
9.     {
10.         description = GetComponent < Text > ();
11.         SetText();
12.     }
13.
14.     void OnEnable()
15.     {
16.         if (progressive) {
17.             TypeText();
18.         } else {
19.             SetText();
20.         }
21.     }
22.
23.     void SetText()
24.     {
25.         if (PlayerPrefsManager1.GetSelectedLanguage() == "EN") {
26.             description.text = english;
27.         } else {
28.             description.text = french;
29.         }
30.     }
31.
32.     public void TypeText()
33.     {
34.         SetText();
35.         string temp = description.text;
36.         description.text = "";
37.         StartCoroutine(TypeSentence(temp));
38.     }
39.
40.     IEnumerator TypeSentence(string temp)
41.     {
42.         for (int i = 0; i <= temp.Length; i++) {
43.             description.text = temp.Substring(0, i);
44.             yield
45.                 return new WaitForSeconds(0.01 f);
46.         }
47.     }
48. }
```

The first two variables will hold the text to display, one for English and one for French. Other languages can be added if other languages need to be supported. The Boolean

progressive will define if the text should be typed in progressively or if it should be immediately displayed. This effect is mostly aesthetic. The Text variable represent the Text object in Unity.

In the **Awake()** function, the Text object is assigned to the local variable and the **SetText()** function is called, this function will replace the text of the object by the appropriate text depending on the language selected. In the **OnEnable()** function, it will check if the text should be displayed progressively or not. If true, it calls the **TypeText()** function which calls the coroutine **TypeSentence(string temp)**. This coroutine will type each letter with a 0.01 second delay to add a special effect when displaying a long text. Otherwise, it will immediately display the text. It is possible to support more languages by creating a new public variable for the language, adding another conditional statement and implementing a new function for the PlayerPrefsManager.

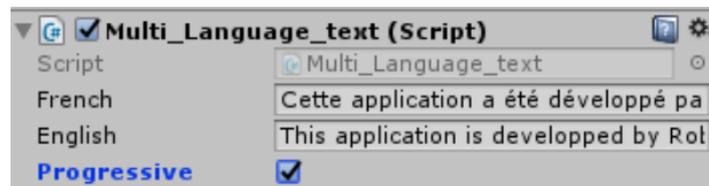


Figure 44 : Example of multi-language text

5.5.5.2 Audio support

The audio file must also support multiple language. After creating the audio files for each language. A new script is added to the Audio Source inside the audio button.

```
1. public class Multi_Language_Audio: MonoBehaviour {
2.
3.     public AudioClip french;
4.     public AudioClip english;
5.     private AudioSource source;
6.
7.     void Awake()
8.     {
9.         source = GetComponent < AudioSource > ();
10.    }
11.
12.    void Update()
13.    {
14.        if (PlayerPrefsManager1.GetSelectedLanguage() == "EN") {
15.            source.clip = english;
16.        } else {
17.            source.clip = french;
18.        }
19.    }
20. }
```

As for the text, two AudioClip variables are declared for each language. These will hold the audio files to play. The source variable will hold the Audio Source of the object linked to the script. The **Awake()** function will assign the Audio Source object to the local variable. In the **Update()** function, it will check which language is selected and will assign the correct clip to play to the audio source.

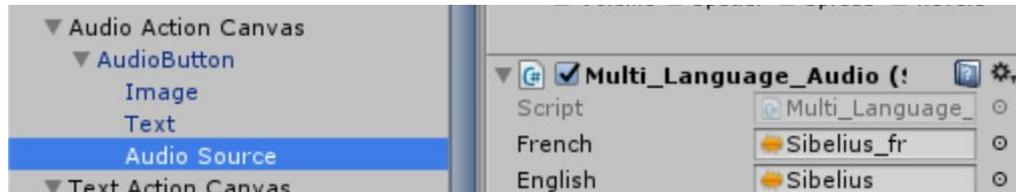


Figure 45 : Example of multi-language audio

5.5.6 Voice command

The voice command tools will be implemented using IBM Watson. After importing the package, a new empty GameObject is created and a new script is assigned to it.

```

1. public class VoiceCommand: MonoBehaviour {
2.
3.     public Button navButton;
4.     public Button audioButton;
5.     static MenuAppear menu;
6.     static Button next;
7.     static Button audio;
8.
9.     private void Awake()
10.    {
11.        next = navButton;
12.        audio = audioButton;
13.        menu = GameObject.FindObjectOfType < MenuAppear > ();
14.    }
15.
16.     public static void LaunchCommand(string command)
17.    {
18.        switch (command) {
19.            case "next ":
20.                next.onClick.Invoke();
21.                break;
22.            case "about ":
23.                audio.onClick.Invoke();
24.                break;
25.            case "menu ":
26.                menu.DisplayMenu();
27.                break;
28.        }
29.    }
30. }
  
```

The two first public variables will hold the buttons created in the scene. The three last variables are declared static because the main function is a static function. In the **Awake()** function, the public variables are assigned to static ones and the menu is assigned using the **FindObjectOfType<MenuAppear>()**. The function returns the first object that

matches the type in between the tags. The static function **LaunchCommand(string command)** receives a string parameter. A switch statement will compare the value of the parameter and will call the corresponding function if the statement returns true. The **onClick.Invoke()** function simulates a click on the button so the result is the same as if it was clicked by the user.

Finally, on the same empty `GameObject`, the example script `ExampleStreaming` from IBM Watson is added to it. This script is modified to use the **LaunchCommand()** function. Lines 175 to 177 are modified.

```
1. string text = string.Format("{0}", alt.transcript);  
2. VoiceCommand.LaunchCommand(text);
```

When the microphone hears the user talk, Watson will listen and decipher what they said. The result is found in the `alt.transcript` variable which is then assigned in the `text` variable. This text is then passed in the **LaunchCommand()** function and if the text is recognized, the action defined in the `VoiceCommand` script will execute.

5.6 Results

All the features have now been integrated into one final application and the prototype has been tested by individuals other than the author. In this chapter, we will discuss the approach we used for the test, the testers and the questions we have asked them. Finally, we will synthesize the feedback received.

5.6.1 Approach

To analyze the results, we will use a qualitative approach. The reason for using a qualitative approach is because we want to develop the application to be used naturally by the users, so personal opinions and comments of the testers will be analyzed. This way, testers can also recommend improvements that we did not consider during the development process. We also want to define which interaction is the most immersive and the reason of their choice. Another reason for choosing this approach is because we own one headset, so we can only conduct the test one person at the time. The cost to buy other headsets would be too great if we wanted to take a quantitative approach.

5.6.2 Test users

As we are using a qualitative research, the sample needed is much smaller than a quantitative research. It is important to have various profiles to obtain thorough results. We will be conducting the full test on 11 people.

The first tests were done with a family with 4 children that are under 13 years old, this family has no relation to the IT field and has never tried virtual reality before. The remaining tests were conducted on students at HES-SO, 4 out of 5 students are currently studying in the IT field and the other one is studying business economics.

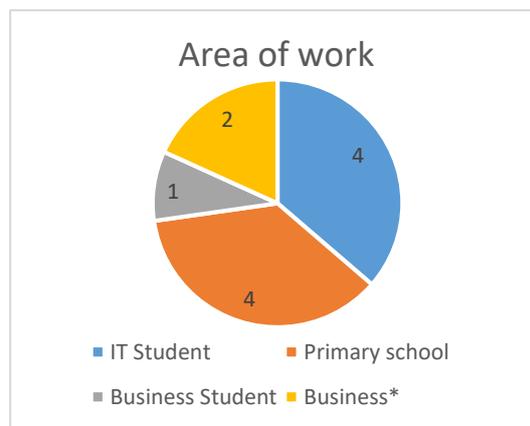


Figure 46 : Division of tester by work area

The family has never tested a virtual reality application, but all the students did test a VR application at least once. One of the IT students possesses an HTC Vive, so consumes regularly VR content and develops VR applications. Outside the scope of this research, the application was also tested by 3 other people during the Geneva Health Forum 2018. The author was running a stand to showcase another VR application and 3 people were curious to test the application as well. During these tests, only the comments and feedback were taken into account, they are not registered in the survey as they did not want their name to be mentioned. Their backgrounds are focused around the health industry and they have all heard about virtual reality, two of them are currently collaborating in a VR project.

5.6.3 Questions

The survey is taking place in 3 phases, the first part is related to the profile of the tester, the second relates to the experience of the testers after testing the application and the last phase relates to the general feedback.

The questions were only asked once the tester has tried the prototype. The testing proceeded as following:

- Greetings.
- Presentation of the thesis.
- Presentation of the prototype.
- Execution of the test using the VR headset, smartphone and audio headset.
- Discussion and questions.
- Thanking.

The questions were the following:

Phase 1: Profile

- First name.
- Age group.
- Area of work.
- Prior use of VR.
- Headset owned.

Phase 2: Prototype

- I enjoyed using the application.
- I felt immersed in the application.
- It was easy to use the application.
- Which was more immersive, text or audio?

Phase 3: General feedback

- Any remarks for improvements.
- Any nausea or after effects.
- Any preference between pictures of videos?

After the questions were answered, a brief and open discussion with the tester about the thesis and on various topics related to virtual reality started, mainly regarding the advantages and benefits of VR technologies.

5.6.4 End results

The feedback was positive, all users enjoyed their play through even if some of them did feel nauseous after a moment (3 people). All testers stated that they enjoyed using the application whether it was their first virtual reality experience or not. A similar result has been obtained when inquired about the immersive nature of the application, they have played along and claimed that they felt immersed inside the application.

Regarding the information, all of them decided that the audio format is more immersive, but having the text displayed is also appreciated as a support. Listening to the audio was compared to having a personal guide with them. Some recommended having the text display bit by bit as a subtitle, this feature could be implemented in a future project. Most have notified that the voice in the audio file is not regular and feels very robotized which makes it less immersive. They were informed that future applications would use more sophisticated voice software or will use a recording of a real person, making them agree that audio format would be more immersive in this case. Some noted that the text format does hide a big amount of the view which is less immersive.

Testers could also give back general feedback about the application. Comments were often repeated and some of them were already taken into account and updated in the application. Most of the important comments can be regrouped into this list:

- Video is better than pictures (however, some did not notice the pictures).
- A bit painful for the eyes (after a while).
- Quality of the video needs to be improved (resolution and camera placement).
- Need better word recognition software and support of other languages.
- Background audio not at the same level.
- Designs need to be renewed.
- Audio voice needs to change and sound more human.
- Strange font when using special character.
- Size of the UI

To summarize, the main point of improvement is the quality of the video and audio. These are mostly hardware related, they can be resolved by using a better 360° camera and a stereoscopic microphone to register sound in all direction. Regarding the voice recognition tool, IBM Watson does not support recognition of other languages, another tool would be required to achieve this as well as better word recognition; a possible alternative would be the Google Speech API. Regarding the audio voice, it was recorded from a free text-to-speech converter, but it did not support French, so two converters were used. The issues can be resolved using a better text-to-speech software which is often a paid subscription or by recording a real person. The designs and fonts are easily updated. Not many improvements can be done to decrease the after effects of using the application as it depends mostly on the individual.

With these results, we demonstrated that it is possible to develop an immersive VR application without any prior knowledge about virtual reality development. Testers agree that the application was enjoyable and that they felt immersed in the virtual world. We also identified that hearing the information is more immersive than having a panel to read.

6 Discussion

We arrive at the end of this research. This chapter will synthesize the work done in this thesis. We will also discuss the different problems we encountered during the development process as well as further development possibilities.

6.1 Summary

As mentioned during this report, virtual reality is only at its beginning. It will be exciting to see the evolution of this technology. At this stage, mobile VR experiences are rather limited, especially when using a 360° camera, and the level of immersion depends on the equipment used. Smartphones are very versatile but can't handle too powerful VR application. Nevertheless, they are the most accessible form of virtual reality and this thesis demonstrated how anyone, with limited knowledge and resources, could set-up their own VR application for leisure or for business.

The technology will greatly improve over time and with the arrival of fully standalone headsets like the HTC Focus or the Oculus Go, portable and interactive VR will become more accessible. At the same time, screen resolution will most likely see an increase in pixel resolution and hopefully reach eye resolution.

6.2 Conclusion

The main goal of this thesis was to prove if it was possible to create a proprietary and immersive VR application. The first part of this thesis focused on gaining a better insight on virtual reality technology and how it is being used. The second part demonstrated how the application was developed with the tools acquired at the time.

To evaluate the final result, 11 people tested the final application by themselves under the author's supervision. They were independent and used the application as they pleased. After their play through, they were asked a few questions about their overall experience. The testers' feedback was mostly positive stating it was an enjoyable experience, the tester's background did not really affect the overall experience. The testers also stated that audio feedback was better and more immersive than having the text displayed in front of them.

Given the limitations of this study, the author couldn't dive too deep as it is a wide subject. However, this thesis has allowed us to learn more about the technology and what it takes

to develop a basic VR application. The evolution of VR technology has increased exponentially, and the different industries are getting interested to invest in it. It can be deduced that this technology will keep improving over the next years.

6.3 Quality of the results

The author considers the final product to be what was initially imagined. Virtual Reality is a technology with great potential. This thesis has demonstrated that it is possible to create a proprietary VR application without using another service. The created application includes all the features defined at the beginning of the thesis as well as some new ones such as a voice command tool and the support for multiple languages. The code has often been refactored to optimize the application as much as possible and Unity sources have been prioritized when searching development information. Regarding the theoretical part, the author's research was done by using web sources to obtain recent intel; sources were often double checked on a different website, so the author considers the source to be correct. Research papers and published books were also included in the research but in lesser quantity as many of them were outdated (before 2015, the year where VR started to catch interest again).

This application has been developed by a single person and the different choices in features and assets can be criticized since they are subjective. Regarding the conducted survey, the testers were all close to the author, so the legitimacy of their answers can also be questioned as all of them were positive.

6.4 Problems encountered

Author considers the final product as being a success but wouldn't recommend this method for other businesses looking to creating a simple VR application like this one. Using Unity has proven more challenging than intended even while using the stable release of the engine. Some of the challenges faced are related to the low amount of documentation regarding Unity VR, and the limited amount that exists is often outdated or is not compatible with the latest version of Unity due to frequent updates. Another challenge was the Unity Engine itself, using this framework often caused strange behavior (e.g. the buttons are not clickable anymore even if no changes were made to them) and some features that are fully functional in the Unity emulator do not work on a smartphone or vice-versa. Building the application to a smartphone often caused the Unity Engine to crash and sometimes it could not build the smartphone version. All these problems did take a considerable amount of time to solve.

6.5 Evaluation of the thesis process

This chapter will discuss the author's methodology and evaluate what he has learned during the thesis process.

6.5.1 Methodology

In order to develop the application, a continuous research process was executed. When implementing a feature, a prototype was first tested without the VR side until it worked successfully. When the implementation was working in a standard 3D or 2D environment, it was then ported to the main project and adapted. When faced with a feature with no knowledge on how to start, Unity documentation and forums were browsed to find a potential starting point. If the research was unsuccessful, time was spent on learning more about the engine by following online tutorials from websites like Udemy or Pluralsight which offer introductory content to advanced principles.

6.5.2 Evaluation of self-learning

The theoretical research was time consuming but very beneficial as it was an opportunity to learn more about the technology. The most challenging part was the development of the application, some of the challenges were explained in the previous chapter (6.4). Refactoring the application was a recurrent process as new ideas and concepts came along when further developing the application, causing the author to spend more time on the same functionality. The author also had troubles writing the report as it was difficult to properly structure and formulate his content.

If the process were to be redone again, the same tools would be utilized but the methodology would change. Tutorials would have been prioritized first to have a better understanding of Unity's development framework and features. These would have been implemented using the Test-Driven Development technic to avoid refactoring all the time.

6.6 Project management

Regarding project management, every task was executed in time. The thesis started end of January 2018 and concluded in April 2018. The order of the tasks was slightly modified as the knowledge acquired at the time allowed to more easily execute another. Three meetings were held with the thesis advisor: the initial meeting on the first week, another

on February 26, 2018 and the last one on April 20, 2018 to demonstrate the application. The rest of the correspondence was done by e-mail for feedback and organization.

The author considers the application to be working as intended and components will very likely be reused in future projects requiring similar features. Designs can be rapidly changed, and the code is easily updated to fit different scenarios. Experience acquired using the framework is already being used in the creation of a full 3D VR application. The author has also learned more about the emergent XR technology and mostly, learned how to better manage his project.

6.7 Ethical viewpoints

In future professional projects, the use of a more advanced 360° camera will require caution. If the application is to be commercialized, there must not be anybody on the picture unless they have given their permission. With our camera, the footage is not clear enough to recognize a face but with a better camera it could be the case. Hiding the face is another option but will eventually break the immersion of the user. The same caution must be exercised with car plates.

6.8 Further development

One of the better areas of improvement would be to invest in more professional 360° gear (camera and microphone). This improvement is already possible as more advanced cameras exist. Further plans can involve improvements in the field of interactions by adding additional inputs or improving the voice recognition with a tool that supports languages other than English, like the Google Speech API. The true potential of virtual reality technologies lies within a 3D environment with all 6DOF included. A 3D VR application using the HTC Vive is being planned and a professional VR project similar to this application is currently being developed by the author.

7 Table of Figures

Figure 1 : Stereoscopic 3D image (tpeimageanimee, 2016).....	4
Figure 2 : Sutherland's HMD, The Sword of Damocles (Computer graphics, 2018)	5
Figure 3 : Example of a flat 360° picture	7
Figure 4 : Example of augmented reality (Bell, 2017).....	8
Figure 5 : Teardown of the Oculus Rift (How It Works Team, 2016).....	9
Figure 6 : Google Cardboard V2 (Google Cardboard V2, s.d.).....	10
Figure 7 : Samsung Gear VR with controller (Gear VR, s.d.)	10
Figure 8 : Desktop requirements for VR (Nvidia).....	12
Figure 9 : Oculus Rift with touch controllers (Oculus Rift, s.d.).....	13
Figure 10 : HTC Vive (Verkkokauppa, s.d.).....	14
Figure 11 : Room-scale setup for HTC Vive.....	15
Figure 12 : Play area for HTC Vive and Oculus Rift (Lang, 2016)	15
Figure 13 : PSVR with a PS4 Pro (Verkkokauppa, s.d.).....	16
Figure 14 : All 6 WMR headsets (Microsoft, s.d.)	17
Figure 15 : Headset sales in Q3 2017 (Alto, 2017).....	18
Figure 16 : Myself testing Birdly	19
Figure 17 : Unity's logo (Unity, s.d.)	24
Figure 18 : Unreal's logo (Unreal Engine, s.d.).....	25
Figure 19 : CryEngine's logo (CryEngine, s.d.)	25
Figure 20 : Movement button icon.....	28
Figure 21 : Audio button icon	28
Figure 22 : Text button icon	28
Figure 23 : Hierarchy of our template scene	34
Figure 24 : Video player sphere.....	34
Figure 25 : Video player in a sphere	35
Figure 26 : Build settings	36
Figure 27 : Public variable in editor.....	38
Figure 28 : Our button's animator state.....	41
Figure 29 : EnterHover animation panel.....	41
Figure 30 : Animated button.....	42
Figure 31 : On Click example.....	43
Figure 32 : Audio button On Click example.....	44
Figure 33 : Example of button and panel overlaying	45
Figure 34 : Text button On Click example	45
Figure 35 : Main menu panel with vertical group component	46
Figure 36 : Country selection	46
Figure 37 : About panel.....	47

Figure 38 : Layout of the setting panel	49
Figure 39 : Menu Controller public variables	52
Figure 40 : Menu switch On Click example	52
Figure 41 : Player object with the panel	53
Figure 42 : The rotation of the cylinder.....	54
Figure 43 : Menu Pop-up	54
Figure 44 : Example of multi-language text.....	56
Figure 45 : Example of multi-language audio	57
Figure 46 : Division of tester by work area	59

8 References

- _Birdly® - The Ultimate Dream of Flying.* (n.d.). Retrieved February 24, 2018, from Somniacs: <http://www.somniacs.co/media.php>
- 360 degree video.* (n.d.). Retrieved February 17, 2018, from Wikipedia: https://en.wikipedia.org/wiki/360-degree_video
- Aaron, P., Zeller, M., & Wojciakowski, M. (2017, December 10). *Inside-out tracking.* Retrieved February 24, 2018, from Windows Dev Center: <https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/tracking-system>
- Admin_renaud. (2016, February 17). *Tour d'horizon de la vidéo 360 : définition, usage, création, ...* Retrieved February 15, 2018, from Realite-Virtuelle.com: <https://www.realite-virtuelle.com/tout-savoir-sur-video-360>
- Adventures-mat.* (n.d.). Retrieved March 6, 2018, from Adventures-mat: <https://www.adventures-mat.com/>
- Alto, P. (2017, December 27). *Media alert: Virtual reality headset shipments top 1 million for the first time.* Retrieved February 20, 2018, from Canalys: <https://www.canalys.com/newsroom/media-alert-virtual-reality-headset-shipments-top-1-million-first-time>
- Animation Parameters.* (n.d.). Retrieved February 6, 2018, from Unity Documentation: <https://docs.unity3d.com/Manual/AnimationParameters.html>
- Ateo. (2017, October 13). *Unity Asset Store, 1.7.2017.* Retrieved February 20, 2018, from GazeClick for GoogleVR: <https://assetstore.unity.com/packages/tools/input-management/gazeclick-for-googlevr-77469>
- Bell, K. (2017, September 24). *Download this: Ikea's AR app lets you preview furniture before you buy.* Retrieved March 7, 2018, from Mashable: <https://mashable.com/2017/09/24/download-this-ikea-place-ar-kit-app/#pvNrKP78siq4>
- Boger, Y. (2017, April 10). *Understanding Pixel Density & Retinal Resolution, and Why It's Important for AR/VR Headsets.* Retrieved February 18, 2018, from RoadToVR: <https://www.roadtovr.com/understanding-pixel-density-retinal-resolution-and-why-its-important-for-vr-and-ar-headsets/>
- Bradshaw, T. (2017, December 7). *Sony dominates sluggish VR headset market.* Retrieved February 20, 2018, from Financial Times: <https://www.ft.com/content/bff48624-daf5-11e7-a039-c64b1c09b482>
- Buckley, S. (2015, May 19). *This Is How Valve's Amazing Lighthouse Tracking Technology Works.* Retrieved February 19, 2018, from GIZMODO:

<https://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768>

Burt, M. (2018, January 2). *Virtual reality has added a new dimension to theme park rides — so what's next for thrill-seekers?* Retrieved March 6, 2018, from The conversation: <http://theconversation.com/virtual-reality-has-added-a-new-dimension-to-theme-park-rides-so-whats-next-for-thrill-seekers-89222>

Buttons in Canvas render mode "world space" not working. (2015, December 30). Retrieved February 21, 2018, from Unity Answers: <https://answers.unity.com/questions/1119289/buttons-in-canvas-render-mode-world-space-not-work.html>

Cimetiere, J. (2018, February 15). *ProBuilder joins Unity offering integrated in-editor Advanced Level Design.* Retrieved February 20, 2018, from Unity Blog: <https://blogs.unity3d.com/2018/02/15/probuilder-joins-unity-offering-integrated-in-editor-advanced-level-design/>

Component.GetComponentInChildren. (2018, March 3). Retrieved March 13, 2018, from Unity Documentation: <https://docs.unity3d.com/ScriptReference/Component.GetComponentInChildren.html>

Computer graphics. (2018, February 7). Retrieved February 14, 2018, from Wikipedia: https://en.wikipedia.org/wiki/Computer_graphics

CryEngine. (n.d.). Retrieved February 21, 2018, from CryEngine: <https://www.cryengine.com/>

Degrees of freedom. (2016, April 27). Retrieved February 25, 2018, from XinReality: https://xinreality.com/wiki/Degrees_of_freedom

Digi-Capital. (2018, January). *Record over \$3B AR/VR investment in 2017 (\$1.5B+ in Q4).* Retrieved February 16, 2018, from Digi-Capital: <https://www.digi-capital.com/news/2018/01/record-over-3b-ar-vr-investment-in-2017-1-5b-in-q4/#.Wof27eexVhG>

Duel : Oculus Rift vs HTC Vive. (2016, June 9). Retrieved February 18, 2018, from Les Numériques: <https://www.lesnumeriques.com/casque-realite-virtuelle/duel-oculus-rift-vs-htc-vive-a2743.html>

Edwards, B. (2015, August 21). *Unraveling The Enigma Of Nintendo's Virtual Boy, 20 Years Later.* Retrieved February 14, 2018, from Fast Company: <https://www.fastcompany.com/3050016/unraveling-the-enigma-of-nintendos-virtual-boy-20-years-later>

Espineli, M., & Thang, J. (2016, October 13). *PlayStation VR: Everything You Need to Know About PSVR.* Retrieved February 23, 2018, from Gamespot: <https://www.gamespot.com/articles/playstation-vr-everything-you-need-to-know-about-p/1100-6443357/>

- FadeInOut*. (2012, November 9). Retrieved March 19, 2018, from Unity Community Wiki:
<http://wiki.unity3d.com/index.php?title=FadeInOut>
- Feloni, R. (2017, June 1). *Walmart is using virtual reality to train its employees*. Retrieved March 6, 2018, from Business Insider Nordic:
<http://nordic.businessinsider.com/walmart-using-virtual-reality-employee-training-2017-6?r=US&IR=T>
- Gear VR*. (n.d.). Retrieved March 5, 2018, from Samsung.
- Gidley, S. (2017, October 22). *Virtual reality: Tourism firms use VR to attract visitors*. Retrieved March 7, 2018, from BBC: <http://www.bbc.com/news/uk-wales-41635746>
- Google Cardboard V2*. (n.d.). Retrieved March 4, 2018, from Google Store:
<https://store.google.com/>
- Haar, R. t. (2005, June). *Virtual Reality in the Military: Present and Future*. pp. 1-2. Retrieved March 6, 2018, from
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.3048&rep=rep1&type=pdf>
- Hardawar, D. (2017, April 18). *Samsung's Gear VR controller makes mobile VR more immersive*. Retrieved February 16, 2018, from Engadget:
<https://www.engadget.com/2017/04/18/samsung-gear-vr-controller-review/>
- How It Works Team. (2016). *How does virtual reality work?* Retrieved February 16, 2018, from How It Works: <https://www.howitworksdaily.com/how-does-virtual-reality-work/>
- Inc., V. (2016, September 26). *Why do all the 360 VR videos today look so pixelated?* Retrieved February 16, 2018, from Medium: <https://medium.com/visbit/why-do-all-the-360-vr-videos-today-look-so-pixelated-b1ab3cba6f95>
- Insta360 One*. (n.d.). Retrieved February 19, 2018, from Insta360:
<https://www.insta360.com/product/insta360-one/?inspm=77c1c2.6666cd.0.0>
- Jackson, D. B. (2015, July 2). *How Does Virtual Reality Work?* Retrieved February 12, 2018, from MarxentLabs: <https://www.marxentlabs.com/how-does-virtual-reality-work-part-i-of-2/>
- Koolonovich, N. (2018, March 3). *Oculus Rift Takes The Lead In Steam Hardware Survey*. Retrieved March 7, 2018, from VR focus: <https://www.vrfocus.com/2018/03/oculus-rift-takes-the-lead-in-steam-hardware-survey/>
- Lang, B. (2016, December 5). *Oculus Rift and HTC Vive Roomscales Compared*. Retrieved February 19, 2018, from RoadToVR:
<https://www.roadtovr.com/oculus-touch-and-htc-vive-roomscale-dimensions-compared-versus-vs-visualized/>

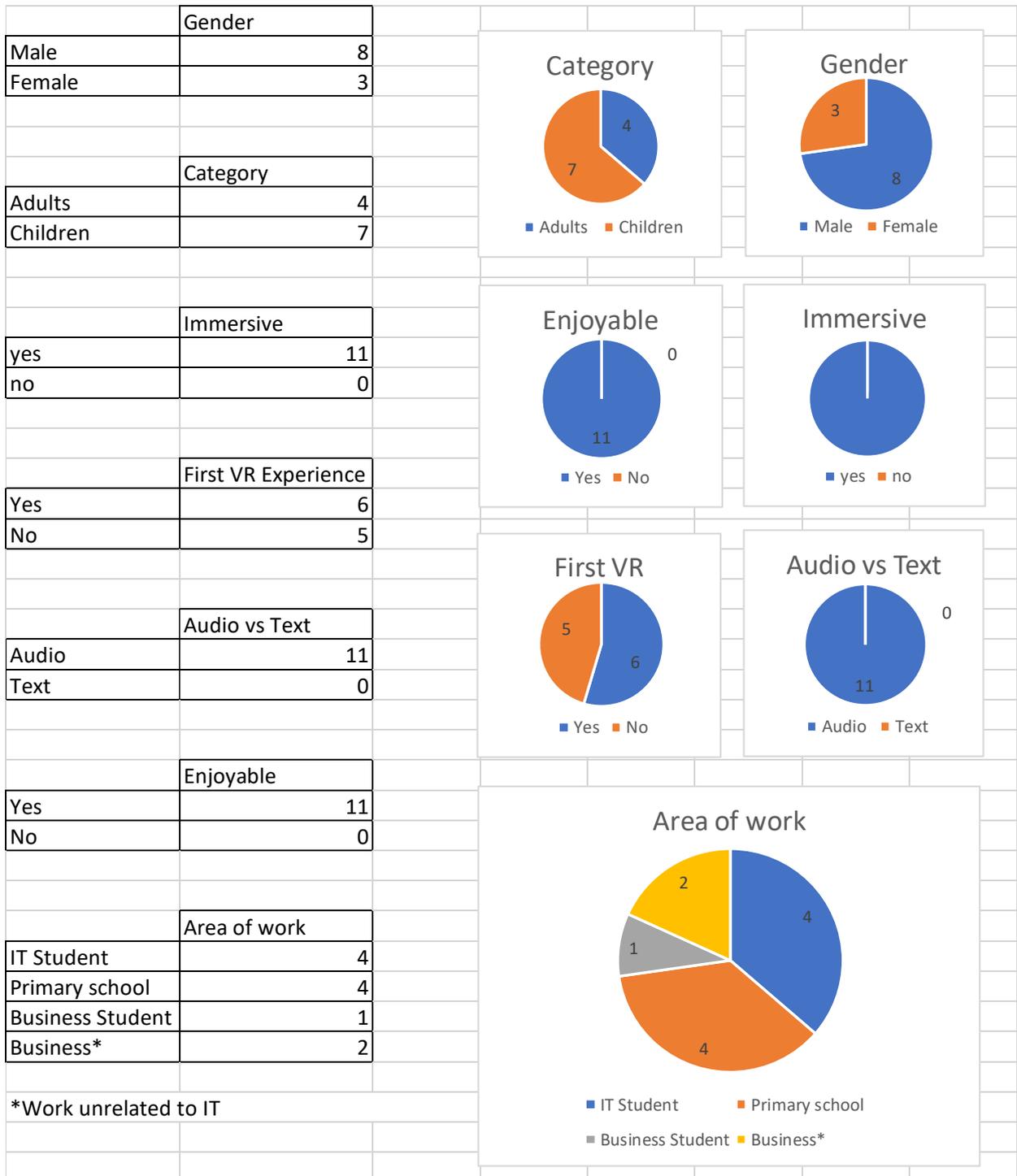
- Mark. (2017, January 10). *5 Common Misconceptions About Virtual Reality*. Retrieved February 13, 2018, from VU Dream: <http://www.vudream.com/common-misconceptions-about-virtual-reality/>
- Matt Swider, M. F. (2016, 01). *HTC Vive vs Oculus Rift: which VR headset is better?* Retrieved from TechRadar: <http://www.techradar.com/news/wearables/htc-vive-vs-oculus-rift-1301375>
- Maubon, G. (2018, February 11). *La réalité augmentée au CES 2018*. Retrieved February 14, 2018, from Association de promotion de la Réalité Augmentée: <http://www.augmented-reality.fr/2018/02/la-realite-augmentee-au-ces-2018/>
- Mercer, C. (2017, January 30). *How virtual reality, augmented reality and mixed reality could be used in education*. Retrieved March 6, 2018, from ComputerWorld UK: <https://www.computerworlduk.com/mobile/how-vr-ar-mr-could-be-used-in-education-3653710/>
- Mesko, D. B. (n.d.). *Virtual Reality Will Change The Healthcare Experience*. Retrieved March 6, 2018, from The Medical Futurist: <http://medicalfuturist.com/virtual-reality-in-healthcare/>
- Microsoft. (n.d.). *Introducing Windows Mixed Reality*. Retrieved February 24, 2018, from Microsoft: <https://www.microsoft.com/en-us/windows/windows-mixed-reality>
- Nield, D. (2016, March 29). *How Oculus Rift works: Everything you need to know about the VR sensation*. Retrieved February 16, 2018, from Wareable: <https://www.wareable.com/oculus-rift/how-oculus-rift-works>
- Nvidia. (n.d.). *System Recommendations for VR*. Retrieved February 18, 2018, from GeForce: <https://www.geforce.com/hardware/technology/vr/system-requirements>
- Oculus. (2016, January 30). *Oculus Rift: Step Into the Game*. Retrieved February 16, 2018, from Kickstarter: <https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game?lang=fr>
- Oculus Rift*. (n.d.). Retrieved March 05, 2018, from Oculus: <https://www.oculus.com/rift/#oui-csl-rift-games=star-trek>
- Oculus Rift History – How it All Started*. (2015, May 18). Retrieved February 17, 2018, from Riftinfo: <https://riftinfo.com/oculus-rift-history-how-it-all-started>
- Ouramdane, N., Otmane, S., & Mallem, M. (2009). *Interaction 3D en Réalité Virtuelle - Etat de l'art*. Hermès-Lavoisier. Retrieved 02 2018
- Peckham, M. (2016, October 5). *Sony's Virtual Reality Expert Explains How PlayStation VR Works*. Retrieved February 24, 2018, from Time: <http://time.com/4520123/playstation-vr-virtual-reality-headset/>
- Perception Neuron*. (n.d.). Retrieved March 5, 2018, from Noitom: <https://www.noitom.com/solutions/perception-neuron>
- Pete. (2016, February 24). *Passive VR – what is “passive virtual reality”?* Retrieved February 15, 2018, from 3dspace: <http://3dspace.com/2016/02/passive-vr/>

- Powell, W. (2017, June 26). *Five ways virtual reality is improving healthcare*. Retrieved March 6, 2018, from Independent: <http://www.independent.co.uk/life-style/health-and-families/five-ways-virtual-reality-is-improving-healthcare-a7801006.html>
- Prefabs*. (n.d.). Retrieved 02 21, 2018, from Unity Manual: <https://docs.unity3d.com/Manual/Prefabs.html>
- Quickstart for Google VR SDK for Unity with Android*. (n.d.). Retrieved 02 21, 2018, from Google VR: <https://developers.google.com/vr/develop/unity/get-started>
- Quickstart for Google VR SDK for Unity with Android*. (n.d.). Retrieved February 21, 2018, from Google VR: <https://developers.google.com/vr/develop/unity/get-started>
- Reach into virtual reality with your bare hands*. (n.d.). Retrieved March 5, 2018, from Leap Motion: <https://www.leapmotion.com/>
- Reichhardt, T. (2017, October 5). *Cosmonauts Shoot First 360 Video of a Spacewalk*. Retrieved February 13, 2018, from Air&Space: <https://www.airspacemag.com/daily-planet/cosmonauts-shoot-first-360-video-spacewalk-180965133/>
- Robertson, A. (n.d.). *The ultimate VR headset buyer's guide*. Retrieved February 18, 2018, from The Verge: <https://www.theverge.com/a/best-vr-headset-oculus-rift-samsung-gear-htc-vive-virtual-reality#comparisonable>
- Scenes*. (n.d.). Retrieved February 21, 2018, from Unity Manual: <https://docs.unity3d.com/Manual/CreatingScenes.html>
- Schreier, J. (2016, March 15). *Crytek's Video Game Engine Is Now Free*. Retrieved March 05, 2018, from Kotaku: <https://kotaku.com/cryteks-video-game-engine-is-now-free-1765078659>
- Shamsian, J. (2017, November 26). *China is building a virtual reality theme park that looks like it's straight out of the future — take a look around*. Retrieved March 6, 2018, from Insider: <http://www.thisisinsider.com/virtual-reality-theme-park-in-china-2017-11>
- Smartphones Minimum Specifications*. (n.d.). Retrieved February 17, 2018, from IrisVR: <https://help.irisvr.com/hc/en-us/articles/220199368-Smartphones-Minimum-Specifications>
- Souppouris, A. (2016, March 18). *How HTC and Valve built the Vive*. Retrieved February 18, 2018, from EndGadget: <https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/>
- SpaceX. (2013, October 5). *The Future of Design*. Retrieved March 6, 2018, from YouTube: https://www.youtube.com/watch?v=xNqs_S-zEBY
- Spears, T. (2017, January 15). *ford's virtual reality lab revolutionizes vehicle design process*. Retrieved March 6, 2018, from Designboom: <https://www.designboom.com/technology/ford-virtual-reality-lab-vehicle-design-01-15-2017/>

- Sprint Vector*. (n.d.). Retrieved February 24, 2018, from Survios:
<https://survios.com/sprintvector/>
- Steam & Game Stats*. (2018, February 18). Retrieved February 19, 2018, from Steam:
<http://store.steampowered.com/stats/>
- Stein, D. (2018, February 8). *Mixed Reality (AR/VR) is back in 2018—Investment is poised to flourish*. Retrieved February 16, 2018, from Medium:
https://medium.com/@don_36842/mixed-reality-ar-vr-is-back-and-investment-is-poised-to-explode-in-2018-f7264816a856
- Stimac, B. (2014, July 25). *How to enable Developer Options on your Android phone or tablet*. Retrieved February 22, 2018, from Greenbot:
<https://www.greenbot.com/article/2457986/android/how-to-enable-developer-options-on-your-android-phone-or-tablet.html>
- Studio, D. C. (2017, March 11). *Unity: Fade Between Scenes*. Retrieved February 12, 2018, from YouTube: <https://www.youtube.com/watch?v=iV-igTT5yE4>
- superventure. (2011, June 21). *LookAt in opposite direction?* Retrieved March 17, 2018, from Unity Answers: <https://answers.unity.com/questions/132592/lookat-in-opposite-direction.html>
- Swider, M., & Fitzsimmons, M. (2018, January 24). *HTC Vive vs Oculus Rift: which VR headset is better?* Retrieved February 17, 2018, from techradar:
<http://www.techradar.com/news/wearables/htc-vive-vs-oculus-rift-1301375>
- Takala, T. (2017). Retrieved March 4, 2018, from LECTURE 5: USER INTERACTION IN VR:
https://mycourses.aalto.fi/pluginfile.php/551951/mod_resource/content/1/Aalto-Lecture5-3DUI.pptx.pdf
- Taylor, H. (2017, December 28). *More than one million VR headsets sold last quarter*. Retrieved February 20, 2018, from gamesindustry.biz:
<https://www.gamesindustry.biz/articles/2017-11-28-vr-headset-sales-exceed-one-million-units-last-quarter>
- The Ultimate Guide to Augmented Reality (AR) Technology*. (n.d.). Retrieved February 19, 2018, from Reality Technologies: <http://www.realitytechnologies.com/augmented-reality>
- Tourism Australia. (n.d.). *New research confirms the potential of virtual reality for destination marketing*. Retrieved March 7, 2018, from Tourism Australia:
<http://www.tourism.australia.com/content/dam/assets/document/1/6/y/7/t/2003897.pdf>
- tpeimageanimee. (2016, January 10). *Fonctionnement de la réalité virtuelle*. Retrieved February 13, 2018, from tpeimageanimee2016.wordpress.com:
<https://tpeimageanimee2016.wordpress.com/2016/01/10/fonctionnement-de-la-realite-virtuelle/>

- Tussyadiaha, I. P., & Jia, D. W. (2017). *Virtual Reality and Attitudes toward Tourism*. Hong Kong: Springer International Publishing. Retrieved March 07, 2018, from https://www.researchgate.net/publication/312046962_Virtual_Reality_and_Attitudes_Toward_Tourism_Destinations
- TV, V. 3. (2017, November 7). *How to make a VR 360 Video app within Unity for Android*. Retrieved February 13, 2018, from YouTube: <https://www.youtube.com/watch?v=jqr0jaPKpMA&t=379s>
- Unity. (n.d.). Retrieved January 28, 2018, from Unity: <https://unity3d.com/fr>
- Unity. (2017, September 25). *VR Essentials Pack Demo - Displaying 360 Video In VR and Switching Views 360 [5/7] Live 2017/9/20*. Retrieved February 11, 2018, from YouTube: <https://www.youtube.com/watch?v=POLjK1sM9KY&t=56s>
- Unity for VR and AR. (n.d.). Retrieved March 5, 2018, from Unity3D: <https://unity3d.com/fr/unity/features/multiplatform/vr-ar>
- Unity Technologies. (2016, September 9). *Unity Samples: UI*. Retrieved March 2108, from Unity Asset Store: <https://assetstore.unity.com/packages/essentials/unity-samples-ui-25468>
- Unity Technologies. (2018, February 16). *ProBuilder, 2.9.8f3*. Retrieved February 20, 2018, from Unity Asset Store: <https://assetstore.unity.com/packages/tools/modeling/probuilder-111418>
- Unreal Engine. (n.d.). Retrieved February 20, 2018, from Unreal Engine: <https://www.unrealengine.com/en-US/blog>
- Verkkokauppa. (n.d.). Retrieved from Verkkokauppa: <https://www.verkkokauppa.com/>
- Virtual reality. (n.d.). Retrieved February 13, 2018, from Wikipedia: https://en.wikipedia.org/wiki/Virtual_reality
- Virtual reality. (2018, February 13). Retrieved February 14, 2018, from Oxford Dictionaries: https://en.oxforddictionaries.com/definition/virtual_reality
- Warren, T. (2017, January 17). *Microsoft's Windows Mixed Reality: everything you need to know*. Retrieved February 24, 2018, from TheVerge: <https://www.theverge.com/2017/10/17/16487936/microsoft-windows-mixed-reality-vr-headsets-guide-pricing-features>
- Warren, T. (2017, September 1). *What is Windows Mixed Reality?* Retrieved February 24, 2018, from TheVerge: <https://www.theverge.com/2017/9/1/16232704/microsoft-windows-mixed-reality-headsets-controllers>
- What is Unreal Engine. (n.d.). Retrieved March 4, 2018, from Unreal Engine: <https://www.unrealengine.com/en-US/what-is-unreal-engine-4>

Appendix 2. Statistics of the results



Appendix 3. Unity project

The Unity project is available on an UBS key, the project can be opened using the Unity 3D framework 2017.3.